AFWAL-TR-86-2072

AI GAS TURBINE ROTOR DIAGNOSTICS

B. Aggarwal
J. Tecza
J. Giordano
R. Brunner

Mechanical Technology Incorporated
968 Albany-Shaker Road
Latham, New York 12110

November 1986

Final Report for Period April 1985 - July 1986

Approved for public release; distribution is unlimited

AEROPROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6563

[illegible]
STEPHEN J. PRZYBYLKO
Project Engineer
Components Branch
Turbine Engine Division
Aero Propulsion Laboratory

[illegible]
JACK D. MATTINGLY, Lt Col, USAF
Chief, Components Branch
Turbine Engine Division
Aero Propulsion Laboratory

FOR THE COMMANDER

[illegible]
MICHAEL E. STEFKOVICH, Major, USAF
Deputy Director
Turbine Engine Division
Aero Propulsion Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

A17899[illegible]

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) [illegible] 86-TR-36	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-2072
6a. NAME OF PERFORMING ORGANIZATION Mechanical Technology Inc. — 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Wright Aeronautical Laboratories Aeropropulsion Laboartory (AFWAL/POTC)
6c. ADDRESS (City, State and ZIP Code) 968 Albany Shaker Road Latham, NY 12110	7b. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433-6563
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-2513
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61101F	ILIR	P5	01

11. TITLE (Include Security Classification)
AI Gas Turbine Rotor Diagnostics

12. PERSONAL AUTHOR(S)
B. Aggarwal, J. Tecza, J. Giordano, R. Brunner

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 4-85 TO 7-86	November 1986	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Expert Systems — Artificial Intelligence
Diagnostic Systems — Rotor Diagnostics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

With increased emphasis on improving the reliability and maintainability of gas turbine engines in the U.S. Air Force (USAF) inventory and the need to operate aircraft form austere forward bases, new diagnostic tools are required. Such tools must provide reliable, consistent diagnoses and minimize training requirements for maintenance personnel. Knowledge-based diagnostic systems have the potential to meet these needs by improving the productivity of USAF maintenance personnel, affecting a standardized diagnostic approach at all maintenance facilities, and providing a wider dissemination of the benefits of accumulated USAF experience in gas turbine vibration diagnostics. This report describes the results of a ~~one-year~~ program designed to demonstrate the feasibility of using knowledge-based diagnostic systems for gas turbine engine rotordynamic diagnostics.

Typical rotordynamic faults observed in gas turbine engines were surveyed to assess the extent of knowledge required to diagnose rotordynamic faults. A set of generic

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Stephen J. Przybylko	22b. TELEPHONE NUMBER (Include Area Code) (513) 255-6690	22c. OFFICE SYMBOL AFWAL/POTC

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

rotordynamic faults was selected and a diagnostic strategy was developed for those faults. System configuration for a typical knowledge-based rotordynamic diagnostic system was defined based on the requirements identified during the survey.

The diagnostic concepts developed were demonstrated using a laboratory test rig capable o having faults implanted in it. Five faults were selected: unbalance, misalignment, rub, increased support flexibility and accessory vibrations. The diagnostic logic for the five faults was implemented as a knowledge-based system (KBS) using the HARVEST fault tree analyzer, ~~developed at MTI~~. The KBS was interfaced to a data acquisition system to acquire vibration data for diagnosis. The integrated system was used successfully to diagnose all faults implanted in the test rig.

Preliminary implementation plans were outlined for a prototype system intended to demonstrate the feasibility of and the operational benefits of using knowledge-based rotor diagnostic systems under realistic conditions.

Unclassified

SECURITY CLASSIFICATION OF THIS

ACKNOWLEDGEMENTS

The work described in this report was funded by the Air Force Wright Aeronautical Laboratories, Aero Propulsion Laboratory, under contract F33615-85-C-2513. Mr. Stephen J. Przybylko (AFWAL/POTC) was the Program Manager for the Air Force.

The authors wish to acknowledge the contributions made by other Mechanical Technology Incorporated personnel:

D. Kubish for work on the demonstration Test Rig hardware

E. Seeger for his work on the Data Acquisition System

D. Smith for configuring the computer systems and supporting the HARVEST Fault Tree Analyzer

F. Gillham for providing programming support for HARVEST

Contributions of Som D.S.R. Karamchetty are also gratefully acknowledged.

A-1

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
1.	INTRODUCTION	1
1.1	OBJECTIVES	1
1.2	CONTENTS OF REPORT	2
1.3	SUMMARY OF RESULTS	3
2.	CONSIDERATIONS OF ROTOR DIAGNOSTICS (TASK 2)	5
2.1	INTRODUCTION	5
2.2	GAS TURBINE COMPONENTS AND FAULTS	5
2.3	DIAGNOSTIC SYSTEMS - OVERALL CONSIDERATIONS	13
2.4	DIAGNOSTIC SYSTEMS - THE AVID APPROACH	17
2.5	KNOWLEDGE REPRESENTATION	19
2.6	DIAGNOSTIC STRATEGY FOR A REPRESENTATIVE SET OF ROTOR SYSTEM FAULTS	28
2.7	IMPLEMENTATION ISSUES	40
3.	ESTABLISHMENT OF EXPERT SYSTEM FEATURES (TASK 3)	42
3.1	OVERVIEW OF EXPERT SYSTEMS	42
3.1.1	IMPLEMENTING AN EXPERT SYSTEM	45
3.2	FEATURES REQUIRED OF AN EXPERT SYSTEM FOR ROTOR DIAGNOSTICS	46
3.2	DATA ACQISITION SYSTEM	46
3.2.2	KNOWLEDGE REPRESENTATION SCHEME	46
3.2.3	USER INTERFACE	47
3.3	EXPERT ROTOR DIAGNOSTIC SYSTEM CONFIGURATION	47

TABLE OF CONTENTS (Concluded)

CHAPTER	TITLE	PAGE
4.	DEMONSTRATION OF AI BASED DIAGNOSTIC SYSTEM CONCEPT (TASK 4)	50
4.1	ROTORDYNAMIC TEST RIG	50
4.2	DATA ACQUISITION SYSTEM	53
4.3	ROTOR DIAGNOSTIC EXPERT SYSTEM	55
4.4	SUMMARY	62
4.5	REFERENCES	66
5.	PRELIMINARY SYSTEM SPECIFICATION AND IMPLEMENTATION PLAN (TASK 5)	67
5.1	ASSESSMENT OF USAF GAS TURBINE ENGINE MAINTENANCE NEEDS AND PRACTICES	67
5.2	APPLICATIONS FOR EXPERT DIAGNOSTIC SYSTEMS	69
5.3	IMPLEMENTATION PLAN FOR A PROOF-OF-CONCEPT DIAGNOSTIC SYSTEM	71
5.4	REFERENCES	72

APPENDICIES

A. DESCRIPTION OF DEMONSTRATION TEST RIG

B. TEST RESULTS FOR IMPLANTED FAULTS

C. REVIEW OF EXPERT SYSTEMS

LIST OF FIGURES

FIGURE NUMBER	TITLE	PAGE
2-1	Example of a Diagnostic Rule for an Unbalanced Low Rotor Turbine	21
2-2	Instructions and Queries for System Automation	22
2-3	Example of a Primitive Fault Tree	24
2-4	Example of a Diagnostic Scheme for a Fault Tree System	25
2-5	Overall Diagnostic Strategy	32
2-6	Data Acquisition and Limit Search Tree	33
2-7	First Low Frequency Fault Tree	35
2-8	Second Low Frequency Fault Tree	36
2-9	Accessory and Unknown Faults Tree	38
2-10	High Frequency Faults Tree	39
3-1	Architecture of a Simple Expert System	43
3-2	Rotor Diagnostic System Configuration	48
4-1	Rotordynamic Demonstration System	51
4-2	Schematic of Test Rig	52
4-3	Frequency Ranges Used for Data Acquisition	54
4-4	Examples of Limb Specification in the Tree Definition Language for HARVEST	56
4-5	Vibration Data Table Format	57
4-6	Vibration Limits and Acceptable Critical Speed Ranges for the Demonstration Test Rig	58
4-7	Symptoms of Implanted Faults	59
4-8	General Flow Chart	60
4-9	Screen Display for HARVEST Fault Tree Analyzer	63
4-10	Data Tables for the Combined Unbalance and Rub Fault	64
4-11	Path Through the Fault Tree for the Combined Fault	65

LIST OF TABLES

TABLE NUMBER	TITLE	PAGE
2-1	Typical Gas Turbine Component Problems	9
2-2	Diagnostic System Considerations	14
2-3	Representative Set of Rotor System Faults	30

CHAPTER 1 - INTRODUCTION

1.1 PROGRAM OBJECTIVES

Maintenance of modern high-performance gas turbine engines such as the F100 and F110 requires high levels of training for flight line maintenance personnel to enable them to interpret a large volume of information encompassing a variety of disciplines. The cost of training is high and is increasing rapidly as the complexity of the engines increases. Considerable hands-on experience is required before flight line personnel achieve proficiency levels necessary to implement strategies such as on-condition maintenance. Exhaustive searches or trial and error techniques used by flight line personnel to compensate for the lack of expertise consume valuable manpower and material resources. Engines which may be fixed by replacing a line replaceable module are sometimes removed from the aircraft for base or depot level maintenance, increasing the workload at those facilities and reducing the availability of engines and aircraft.

Experienced maintenance personnel familiar with all aspects of engine vibration diagnostics are scarce and often unable to attend to all the demands placed upon them. Their time is often spent on jobs which are, for them, routine. The cost of maintenance could be reduced and engine availability increased if the experience of the experts is made available to the less experienced flight line personnel. The need for expertise is even more critical when aircraft are deployed away from home base.

The cost of more extensive training to bring the flight line maintenance personnel to expert level is prohibitive. A more cost effective approach is to capture the knowledge of the experts in an expert system and use the expert system to augument the expertise of the flight line personnel. An expert system could assist them in making more thorough and systematic diagnoses, reduce the time taken to make the diagnoses and enable them to make diagnoses they would not be able to make without the additional information provided by the expert system. The benefits of the expert diagnostic and maintenance systems include:

- Improved quality of flight line maintenance service, resulting in better engine availability
- Standardized maintenance procedures

- Reduced training and maintenance costs
- Better usage of "expert" manpower by reducing the "routine" jobs they handle and enabling them to focus on unique problems
- A flexible, expansible system capable of changing as maintenance practices change and as new engines are acquired

The potential benefits of expert systems can be realized fully only by looking at the maintenance system as a whole to determine the optimum way to incorporate expert systems into the maintenance system for USAF gas turbine engines. However, because the expert systems technology is relatively new, a more prudent approach is to prove the applicability of the technology to USAF needs before committing resources for use in the field.

A one year program with Mechanical Technology, Incorporated (MTI) to demonstrate the feasibility of using expert systems for rotordynamic diagnostics was funded by the Air Force Wright Aeronautical Laboratories, Aero Propulsion Laboratory, Wright-Patterson AFB, Ohio (USAF Conctract No. F33615-85-C-2513). The objectives of the program were:

1. Demonstrate the feasibility of using expert system based rotordynamic diagnostics.

2. Develop a preliminary system specification and implementation plan for using expert rotordynamic diagnostic systems for USAF gas turbine engines.

This report describes the work accomplished in the program.

1.2 CONTENTS OF THE REPORT

The extent of knowledge required and other considerations of rotor diagnostics (Task 2) are described in Chapter 2. Features required for an expert rotordynamic diagnostic system are described in Chapter 3. A review of existing expert systems and related technologies is included in Appendix C. The demonstration of expert diagnostic system concepts using a laboratory test rig is described in Chapter 4. The details of the demonstration test rig are given in Appendix A and diagnosis of each of the implanted faults in the test rig is described in detail in Appendix B. The report concludes with the description of a preliminary system specification and implementation plan in Chapter 5.

1.3 SUMMARY OF RESULTS

The extent of knowledge required for gas turbine engines was identified and a fault tree was developed for a set of representative rotordynamic faults observed in gas turbine engines. It was determined that a nonconsultative diagnostic system was the best approach and that fault trees were the best way to represent rotordynamic diagnostic knowledge.

A system configuration for an expert rotordynamic diagnostic system was then developed. The recommended system configuration includes a capability to monitor the performance of sensors in the diagnostic system and detect sensor malfunctions or failures.

The diagnostic concepts developed were demonstrated using a laboratory test rig with the capability to accept implanted faults. Five faults were selected for the demonstration: unbalance, rub, misalignment, increased support flexibility and accessory vibrations. A diagnostic strategy based on the symptoms observed was developed and implemented as an expert system using the HARVEST fault tree analyzer developed by MTI. The expert system was interfaced to a data acquisition to acquire data when needed. The demonstration expert system was used successfully to detect all faults implanted in the test rig.

It was concluded that:

- It is possible to develop a diagnostic strategy for rotordynamic faults in gas turbine engines.
- Placement of sensors is an important issue. If sensors are not placed properly, it may not be possible to detect faults.
- Reliability of sensor data must be assured. Sensor failures or malfunctions must be detected so that the diagnostic system does not report them as possible faults in the engines.
- Data acquisition systems should be designed as a separate, modular sub-system to the diagnostic system. This will allow the expert system

and data acquisition modules to use technologies optimized for their respective needs.

A prototype expert diagnostic sytem is outlined based on an assessment of USAF needs. The prototype system is intended for use at a base level engine repair facility for a modular engine such as the F100. The system will diagnose faults in engines on modules removed from aircraft on the flight line and help maintenance personnel decide whether the engines or modules can be repaired at the base or need to be sent for depot level maintenance. Once the utility of the system has been proven, the diagnostic concepts can be applied to other rotating machinery in USAF inventory such as Auxiliary Power Units.

CHAPTER 2 CONSIDERATIONS OF ROTOR DIAGNOSTICS (TASK 2)

2.1 INTRODUCTION

Aircraft gas turbine engines are highly complex rotating machines which incorporate a variety of components including lightweight casings, multiple rotors, bearings, gears, dampers and accessories. Accordingly, there is a wide variety of mechanical faults which occur with these components and which are manifested by high vibration or unusual vibration patterns. Some of these mechanical faults may be present after overhaul while a great many others will develop while the engine is in service. Detection of these faults is not a reliable process as instrumentation is often minimal and the detection and diagnosis of a given fault often requires an expert. Such experts are scarce. There are few in the overhaul centers and fewer still in the field. Therefore, it would be very advantageous to have an expert system which could identify the existence and diagnose the cause of such faults in either application. Faults not corrected during overhaul or those resulting from incorrect overhaul procedures could be caught during engine testing and isolated so that selective repairs could be made. Problems developing during operation in the field could be quickly located and appropriate action taken, be it field repair, module replacement or return to the overhaul center.

This section describes typical gas turbine components, representative problems and how these faults might be diagnosed. It describes several automated diagnostic procedures including symptom fault matrices, rule bases and fault trees and describes in some detail a set of representative rotor bearing dynamics problems with their specific diagnostics. A representative fault tree is then developed to implement these diagnostics. Finally, a short description is given of expansion and developement needs for fault tree based expert system diagnostics of gas turbine engines.

2.2 GAS TURBINE COMPONENTS AND FAULTS

Gas turbine engines are composed of a variety of rotating components bundled into a very compact package. This limits the amount of vibration and other diagnostic information which can be extracted. The problem of diagnosing faults is further exacerbated by the variety of things which can go wrong as well as the limitations imposed by the scarcity of diagnostic experts and the need to perform diagnoses and repairs quickly. The first item which must be addressed is the instrumentation which is installed on the typical engine. Data normally made available from an engine include:

- Speeds
- Exhaust gas temperature
- Fuel flow
- Oil pressure
- Oil inlet temperature
- Fuel pump inlet pressure
- Inlet air temperature
- Case vibration
  - Velocity pick-ups
  - Accelerometers
- Magnetic plug for oil chip detection

Current techniques for detecting engine mechanical problems using this information include the following:

- Overall vibration readings from case mounted accelerometers or velocity pick-ups
- Debris on magnetic oil plug (manual, automatic)
- Pilot "feel"
- Oil sample debris analysis
- Performance tests
- Mechanical inspections, temperature tests, etc.

Some aircraft do incorporate limited on-aircraft monitoring. Examples of this may be seen in the A10 and in the F15 systems. These are useful but are not necessarily comprehensive. They were not designed for identifying and diagnosing dynamics related problems. In order to develop ideas on how to diagnose dynamics related problems (either with existing instrumentation and procedures, or using additional sensors, equipment or tests), we must first look at the key sub-systems themselves, typical problems and manifestations of those problems.

However, before the mechanical systems are addressed, consideration must be given to problems occurring with sensors themselves and their electronic support equipment. This is a critical issue that must be dealt with in any semi-automated or automated diagnostics approach. Failure of sensors, conditioning electronics or the diagnostic system itself must not be misconstrued as a mechanical engine failure or problem. Therefore, any failures in the data acquisition process must be diagnosed and corrected as soon as possible to assure that only valid data are used as input to the system. Further, failures must be allowed for and tolerated. That is, information from failed sensors must be ignored and that coming from the good sensors has to be used to maximum advantage in the interim. Non-engine related diagnostic system failures can be grouped into four broad categories: Failure of sensors, wiring, conditioning electronics or failure beyond the data acquisition system, that is, of the monitoring system components themselves. These failures can be described in more detail as follows:

1. Sensor failure - Sensors such as accelerometers, velocity pick-ups or thermocouples can fail completely, in which case signal levels are generally low or nonexistent. This can and has resulted in conclusions being drawn that the machinery under scrutiny is running extremely smoothly. However, this type of failure is also very easy to detect. More bothersome is partial failure which may result in an erratic signal, one which is not properly calibrated or one which "comes and goes" with time. Location of this type of failure is often a problem for an expert but can be extremely difficult for an automated system.

2. Wiring failure - Complete failure of wiring can occur through a break in a wire or connector. Partial failure can occur with intermittent interruptions of the signal such as open circuits created by temperature changes or high vibration levels. Substantial noise pick-up can also occur with improperly grounded shields or those which have breaks and can be very difficult to detect.

3. Conditioning electronics failure may involve complete failure, resulting in loss of output signal or partial failure, which is manifested in a variety of ways including calibration changes, erratic output, occasional loss of output and noise pick-up.

4. System failures beyond the data acquisition system can include, for example, monitoring system computer failures, erratic disk reading/writing on problems with analog-to-digital converter electronics

Most of these failures can be detected by regular scanning of the signals going into the system and by execution of proper self-test procedures by the system. In fact, the process of insuring that data are valid can be the subject of an expert sub-system development effort in its own right.

Having progressed beyond the stage of data validity, diagnoses of component problems can now be addressed. These components have been grouped for convenience into the following three categories:

- Rotating
  - Shafting
  - Blading
  - Joints
  - Gears
- Support
  - Bearings
  - Dampers
  - Mounts
- Other
  - Casings
  - Seals

Typical components and their faults are shown in Table 2-1. The table consists of three columns. The first lists the individual components as described above. The second describes some of the major faults associated with the mechanical failure of these components that are observable through vibration/dynamics detection mechanisms. The third column lists some of the typical symptoms. Note that not all of the symptoms listed are associated with all of the faults. Rather, what is given is a cross section of symptoms from which these faults can generally be diagnosed.

Table 2-1 TYPICAL GAS TURBINE COMPONENT PROBLEMS

COMPONENTS	FAULTS	TYPICAL SYMPTOMS
A. SHAFTING/DISKS	. Bowing (Permanent, Thermal)	. Large Synchronous Vibration
	. Misalignment	. Large 2/rev Components
	. Cracking	. Difficulty in Balancing
	. Disk Misalignment/Eccentricity	. Changing Vibration Levels
	. Disk Looseness	
B. BLADING	. Impact Damage	. Large Orbits from Heavy Unbalance
	. Cracking	. Efficiency/Power Loss
	. Breakage	. Blade Pass Frequency Components
	. Erosion	
	. Broken Shrouds/Ties	
	. Deposit Buildup	
C. JOINTS	. Looseness	. Sudden Vibration Level Shifts
	- Wear	. Large Synchronous Vibration
	- Improper Machining	. Instability (rare)
	- Improper bolt-up	. Altered Dynamic Characteristics
	. Lubrication Failure (Splines)	
	. Tooth Breakage	
	. Eccentric Assembly	
D. GEARS	. Surface Damage	. Modulated Vibration at Mesh Frequencies
	. Cracking	. Acoustic Noise
	. Excessive Wear	
	. Misalignment	
	. Support Bearing Faults	
E. BEARINGS	. Corrosion	. Overall Vibration Level Increase (Low, High Frequencies)
	. Fatigue Spalling	. Modulation of High Frequency Signal
	. Wear	. Component Defect Frequencies
	. Mis-Installation	. Wear Debris
	. Race Cracking/R.E. Fracture	
	. External Contamination	
	. Bad Material Lot	

Table 2-1 (Cont'd) TYPICAL GAS TURBINE COMPONENT PROBLEMS

COMPONENTS	FAULTS	TYPICAL SYMPTOMS
F. DAMPERS	. Coking . Failed Seals . Failed Flexures/Supports . Oil Supply or Drain Failure . Foam in Oil . Mis-Installation	. Large Changes in Resonant Characteristics - Critical speed locations - Amplification factors . Additional Frequencies (eg. from looseness . Instability
G. MOUNTS	. Cracking/Structural Failure . Loose Bolts/Components . Thermal or Other Distortion	. Large Changes in Critical Speed Locations . Poor Damper Performance . Seal Rubs
H. CASINGS	. Cracking . Distortion . Misalignment . Guide Vane Problems . Rotating Stall	. Large Critical Speed Location Changes . Poor Damper Performance . Seal Rubs . Blade Pass Frequency Components . Large Amplitudes at Rotating Stall Frequency
I. SEALS (ROTATING & STATIC)	. Wear (Major Problem) - Labyrinth teeth - Abradable material - Seal face wear/cracking (gas path and oil) . Coking . Soak-Back - Damper O-rings - Static seals . Mis-Installation	. Rub Symptoms - Harmonic frequencies - Overall high vibration (Broadband or "Humped") - Acoustic noise - Possible instability . Natural Frequency Shifts . Poor Damper Performance . Efficiency/Power Loss

Problems with shafting and disks often include bowing, (which can be permanent or temporary), misalignment, cracks and loose or eccentric disks. The first two faults are typically characterized by large synchronous and two-per-rev vibration components respectively. The latter three often show up in vibration levels which change for no good reason or in completely unpredictable response to the application of balance weights. A wide variety of problems can occur with compressor and turbine blading. These include damage due to impacts, cracking and breakage, erosion, broken shrouds or ties and build-up of deposits. These tend to cause larger and larger orbits as unbalance levels rise, losses in power or efficiency as blade profiles change. Because of the non-symmetry introduced in flows through eroded or damaged blades, substantial components of vibration will be seen at blade pass frequencies. Rotor joints are typically either loose, for a variety of reasons, or may be tight but are assembled eccentrically. Loose joints produce sudden shifts in vibration level which can often be seen as "step" jumps on speed vs. amplitude response curves. System characteristics are changed because of altered local stiffness and system instability may also result. Eccentric assembly may introduce a large local unbalance producing large synchronous vibration or even create a substantial rotor bow. Gear damage is not a major problem with gas turbines but gears can be subject to a large number of faults including surface damage, cracking, wear, misalignment and secondary damage resulting from problems with support bearings. Gear damage is often perceived as acoustic noise and modulated vibration at the mesh frequencies.

Rolling element bearings, which are almost exclusively used in gas turbines, also have a variety of failure mechanisms including corrosion, spalling, wear, fracture and misinstallation. Most often, problems are detected by large overall vibration level increases in both low and high frequeny ranges and, in extreme cases, by vibration at the defect frequencies of the components. Most problems may be spotted much earlier by scanning the higher frequency ranges (anywhere from the tens to the hundreds of kiloHertz) and looking for modulated signals at these high frequencies. Searching for wear debris is also a very common non-vibration related way to identify rolling element bearing problems. Dampers, and more specifically, squeeze film dampers, can be subject to coking (due to high soak back temperatures), failed seals or supports, lack of oil feed or drain, foam in the oil (which will also affect the bearings) and also misinstallation. Improperly operating dampers result in large changes in critical speed locations and amplification factors at the critical speeds and may also result in rotor system instability if the damping that was provided by the (now misoperating) damper was needed to overcome a major aeroelastic source of excitation. Bearing and damper mounts can suffer major structural failure, be loose from improper assembly or undergo shifts in support locations due to thermal or other distortions. Structural failure or looseness results of course in large changes in critical speed locations and reduced damper performance and any of these faults can result in enough rotor system misalignment to cause seal rubs.

Potential problems with engine casings include cracking or distortion and problems with guide vanes. The first two faults result in large critical speed location changes, poor damper performance and seal rubs, while vibration components at blade pass frequency indicate substantial guide vane problems. In addition, a rotating stall in the compressor produces large amplitudes at the rotating stall frequency. Finally, with regard to seals (both rotating and static) wear is a major problem, along with coking and other damage resulting from high soak back temperatures. Heavy wearing of labyrinth or clearance type seals produces classic rub symptoms including large amplitues at harmonic frequencies and overall high vibration. High soak back temperatures can damage damper O-rings as well as static seals leading to poor damper performance while any major damage to seals in the gas flow path leads to losses in power and efficiency.

To summarize the situation, gas turbines have a wide variety of components and a number of potential faults are possible with each of these components. The symptoms associated with individual faults often overlap, making fault diagnosis confusing. There is also the distinct possibility of multiple faults occuring in a gas turbine which share some of the same symptoms. There are, as well, severe limitations on sensor types, locations and numbers as well as in the methods of data acquisition and there is a real need to ensure that the data being used for diagnosis are accurate.

There is also another consideration, that of whom to inform of the existence of the problem and when to inform that person. This consideration is extremely important, perhaps of equal importance to the diagnosis itself as this is what physically triggers a needed maintenance action.

The above considerations point to several conclusions. First, if diagnosis of gas turbine faults is to be performed in anything like an automated manner, a very substantial knowledge base must be developed and tailored to the specific turbine and its individual components. Second, an expert sub-system must be included to insure that only correct sensor data are used and that the system itself is operating properly. Third, the system must be designed specifically for and integrated into a particular part of the Air Force maintenance procedure, such that its use will become automatic and that its effectiveness will be maximized.

2.3 DIAGNOSTIC SYSTEMS - OVERALL CONSIDERATIONS

This subsection deals with the approach chosen for the implementation of an artificial intelligence based nonconsultative diagnostic system. The approach that will be outlined is only one of many which could be implemented. It is intended to be a broad and illustrative, but nonetheless viable, vehicle with which to give an example of how AI based diagnosis can be implemented, and which can be extended to achieve that implementation.

There are certain considerations for an automated diagnostic system, whether artificial intelligence based or not, which are vital for the success of that system. These considerations are much stricter and much more difficult to apply to a nonconsultative system than to a consultative one. This is because, with the former system, a lot of "up front" data acquisition and processing hardware is needed, and questions of data validity must be answered automatically. The system hardware is difficult to add to or to change because of the high level of integration and in general the higher level of performance expected of a nonconsultative system means that more preparation is required. These considerations apply whether the diagnostic system is embedded in a vehicle, is portable or is in a fixed location at an overhaul center. These considerations are given below in Table 2-2 in rough order of priority, although each is very important in its own right.

The first consideration is a thorough understanding of component or system behavior in faults. That is, what can go wrong with components and how are faults detected? Along with this goes an understanding of the way the system will be used, when to tell an operator what is needed and in what time frame

TABLE 2-2

DIAGNOSTIC SYSTEM CONSIDERATIONS:
- AUTOMATED (NOT CONSULTATIVE) SYSTEM -

1. Understanding of:
   - Component/System Behavior
   - Faults
   - Symptoms
   - System Usage
   - Operator Interface Needs
   - Maintenance Cycle
2. Minimal Operator Involvement
   - Sensors
   - Tests
   - Intervention
3. Simplicity/Ruggedness of Hardware
4. Ease of System Alteration, Expansion
5. Tolerance for Multiple Faults, Unclear Data, Diagnostic Component Failures
6. Reasonable Speed of Operation

action must be taken. The system must be easy to interface with the hardware and must be carefully integrated into the maintenance cycle to make sure that it does, in fact, get used and its positive impact maximized.

A nonconsultative diagnostic system must, by its very nature, require a minimum of operator involvement. The system must operate as autonomously as possible and the use of special sensors or tests must be minimized. Any operator intervention, which is necessary for system operation, must be carefully thought out to maximize the efficiency of the overall process and to make the system as "hassle-free" as possible to use.

An automated diagnostic system must employ hardware that is both simple and rugged, for example, a field portable unit dare not be large and cumbersome nor can it be delicate or it won't be used, either because it is to much of a problem to set up or the system itself fails too often. These requirements must be taken to the extreme with any hardware that might be installed on an aircraft, while they can be relaxed somewhat for a fixed location system at an overhaul center.

A diagnostic system must be easy to expand and to change. Development of a diagnostic system is a learning process which is continuous. Typically, a modest start is made and the system is expanded and changed as knowledge about both systems, the monitoring system and the one being monitored, increase. A difficult-to-modify system will not get used for very long because knowledge or hardware changes will quickly make it obsolete.

A monitoring system must be able to tolerate multiple faults, data which are unclear and failures in the diagnostic components themselves. Simply put, it must live in the "real world" where many components can break at the same time and where diagnostic information is rarely clear.

Finally, the system must operate quickly enough such that diagnoses can be made in a reasonable amount of time. Maintenance personnel and operators simply cannot wait forever. If a diagnostic system is not integral with an aircraft, then it must be able to be attached quickly to an engine or a data acquisition system using a simple procedure. The actual time to interpret the data and diagnose faults must be relatively short and a portable system must be easy to disconnect and move on to the next engine or aircraft.

One of the most difficult tasks which must be faced is physically acquiring the knowledge about a mechanical system in order to specify the diagnostics. This is necessary in order to know what data to look for in the first place, to know how the knowledge base should be structured, to be able to design the way in which the diagnostic system should operate and, finally, the knowledge must be physically entered into the knowledge base. Typically sources for this information are:

- In-house expertise
- Repair and overhaul personnel experience
- Manufacturer's Expertise Handbooks
- Analyses for specific engines involved
- Reviews of operating data
- Special tests

It it also highly desirable to "eliminate the middleman". That is, to eliminate the need for a knowledge engineer so that experts can enter their knowledge directly into the knowledge base in order to get a diagnostic system operating and keep it updated as new knowledge is added or changes are made. At the very least, the system should be simple enough to operate and modify such that the skill level required of the knowledge engineer is minimized.

If this can be done, that is, a nonconsultative artificial intelligence based diagnostic system developed and implemented for gas turbine engine diagnostics, there are many benefits which would result. For example, the ability to perform on-condition maintenance would be greatly enhanced.. Defects in engines could be found eariler and isolated more quickly. This leads to less expensive and less extensive repairs. For faults involving failed components such as accessories, failed components could be isolated quickly and repaired without major changes to the engine. For modular engines, bad modules could be identified quickly and replaced at the appropriate maintenance level. Even where defects are found for which major engine disassembly is required, overhaul personnel can be guided to look for specific problems in specific components, which eliminates unnecessary work and greatly speeds the repair process.

2.4 DIAGNOSTIC SYSTEMS - THE AVID™ APPROACH

In order to develop and illustrate an approach to AI based diagnostics a typical set of diagnostics is needed. The specific approach which will be used has been implemented in the AVID™. diagnostics system which uses a conventional symptom-fault approach to rotor diagnostics and is implemented on large mainframe-type computers.

Specifically, these systems are located at Tinker AFB, Oklahoma City and Kelly AFB, San Antonio in air logistics overhaul centers. They are set up for diagnosis and balancing of TF30 gas turbine engines in test cells following engine overhaul. The diagnostic algorithms are currently symptom fault matrix (SFM) based and are coded in FORTRAN. For this system, vibrations probes at three locations are used. These locations are at the compressor end, the middle and aft end of the engine. The system acquires data from each of these probes in a given test cell and generates spectra in low and high frequency ranges at prespecified engine speeds. The spectra are reduced and processed off-line and scanned for faults using the SFM technique. If balance weights are needed, the operator is instructed to attach a phase reference marker, retake the data and balance weights to be applied are computed.

Implementation of an SFM type technique means that all spectra must be scanned, all out of limit conditions must be noted and the pattern of the "flags" delineating the out of limit conditions is compared with each pre-stored fault pattern to pick out the individual faults. The specific faults which are sought by this particular system are:

- Unbalanced rotor
  - LP compressor
  - LP turbine
  - HP compressor
  - HP turbine
- LP compressor bowed rotor
- Inability to balance
  - Inaccessible balance plane
- Misalignment

LP rotor
HP rotor

- Increased rotor flexibility
- Seal rub or bearing fault
- Accessory vibration
- Unknown frequency fault

The system has several other functions, such as high vibration alarms, which will not be considered. What will be used is the essence of what specific problems are looked for, what data are needed in order to do the search and the overall flow of data, that is acquisition, reduction and scanning, as opposed to some alternative data flow method.

There are a number of advantages and disadvantages which accrue to a symptom fault type approach, especially when a scientific type language is used for the programming. In particular, the approach is fairly easy to implement as diagnostics involve a series of logic statements within the program and possibly a series of table look-ups and comparisons. It is very easy to diagnose similar engine types since specific frequencies and vibration level limits can be stored in data files and accessed by the diagnostic routines. However, several major disadvantages are involved in this approach. For example, a great amount of difficulty is involved with adding new faults, changing the overall diagnostic strategy or accommodating many diverse faults. Major reprogramming of diagnostic algorithms may be needed, along with restructuring of the parameter data files. If the programmer updating the software is not the same programmer who wrote it originally, a substantial amount of relearning of the code is required. In addition, all diagnostic tests must be performed all the time. This can be extremely cumbersome where a large number of diagnostic tests are involved. The decision path taken to arrive at a specific diagnosis may not be very clear as it can be buried deeply in the computer code, with no way to trace the specific decisions made during a diagnostic run. To overcome some of the problems involved when diagnostics are coded with the rest of the system (becoming an integral part of the software) some diagnostic system implementations leave "holes" for certain classes of faults. That is the user may be able to fill in alert or alarm levels and also a fault identity table which relates alert levels to specific faults. However, this is very restrictive and any substantia modification requires recoding of the software.

2.5 KNOWLEDGE REPRESENTATION

In implementing an artificial intelligence based diagnostic system, the first and most fundamental decision which needs to be made is how to structure the knowledge which the system is to contain. Currently, there are two major choices for this knowledge structure: representation of knowledge in a series of rules (rule base) or in the form of a decision tree (fault tree). Each of these representations can handle the same basic knowledge. They differ, however, in the way the knowledge is structured, the way it is operated on by an inference engine, the way it can be extended and in their interface with the knowledge engineer or expert.

With a rule base approach the knowledge is divided into specific rules, for example "If... then...". The fulfillment of a particular condition implies a specific action. These rules are supplied by an expert and put into the proper format by a knowledge engineer. Once a set of rules is formed into a knowledge base, an inference engine operates on the rules as a whole, sorts through them and attempts to find the most efficient path to a conclusion. This strategy takes advantage of the fact that experts do tend to think in terms of rules, therefore it would appear to be a relatively straightforward process in extracting rules and loading them into a knowledge base. Expanding the knowledge base is likewise simple in principle: one merely adds additional rules.

With the fault tree approach, knowledge is represented as a block diagram or decision tree. The expert supplies the individual blocks of knowledge and also defines the pathways to conclusions or diagnoses. The task of the inference engine is to follow various pathways in order to seek the best conclusion(s). The power of the fault tree approach lies in the fact that, while diagnostic experts may think in terms of rules, they frequently organize and visualize their knowledge in terms of a fault tree model. The fault tree tends to be a more highly structured approach in that the expert has to define the decision points and paths as well as the knowledge itself. However, like rules, the blocks or limbs of a fault tree do not represent executable code and therefore can be placed in the knowledge base in any order. Adding knowledge consists of adding additional limbs and rearranging the interconnections to incorporate these limbs. Because the fault tree approach is a very visual approach, the use of a highly trained knowledge engineer to set up the knowledge base is not as critical and the identification of such things as contradictory rules and endless loops in the diagnostic procedure is not as difficult.

The most striking difference between rule based and fault tree approaches appears to be that, while rules can be evaluated by the inference engine in any order, fault tree limbs must be evaluated in the order previously defined by the expert. However, it should be noted that specific groups of rules may very well need to be evaluated in specific orders and this is automatically implied in the rule base by the expert. Further, for a nonconsultative system, adding knowledge either to a rule base or to a fault tree diagnostic system may mean a substantial restructuring of the data acquisition scheme or the evidence data base, either of which can be a major undertaking.

To illustrate the application of both approaches and the level of detail that must be addressed for a nonconsultative diagnostic system, the following example is offered. The system is to determine if, given a certain amount of data, a fault in a gas turbine is an unbalanced low rotor turbine. A set of rules to evaluate the available data and determine whether this fault exists might look like those in Figure 2-1 (although the specific implementation is arbitrary). These rules determine that the peak value of synchronous vibration from a certain probe is within the expected band of one of the critical speeds and if at the same time its level is large enough to be of concern and there does not seem to be much vibration at frequencies other than synchronous, then an unbalanced low rotor turbine is the probable fault. As far as a diagnostician would be concerned, these rules are really not all that complicated. They in fact would be easy to implement in a consultative (question/answer) form, where the user scans through the data and simply answers the questions from the system. However, the situation is much more complicated for a nonconsultative or automated diagnostic system. As an example, the actions which only the first two rules involve are shown in Figure 2-2. These commands direct the system to retrieve the proper data, to interpret that data, to find the location for the speed peak in the appropriate range and the limits for the location of that speed peak for the appropriate probe. Then the system must test whether the speed corresponding to the peak is within the correct band and return the results to the system.

1. IF Speed of Synchronous Peak is Within L1(N1) Band
2. AND Probe is V3
3. AND N1 Synchronous Amplitude is Greater than Warning Limit
4. AND Twice per Rev Amplitude is Less than Synchronous Amplitude
5. AND No Broad Band Noise Exists
6. THEN Fault is Unbalanced Low Rotor Turbine

Figure 2-1

Example of Diagnostic Rule for Unbalanced Low Rotor Turbine

EXAMPLE: STATEMENTS 1 AND 2

- Retrieve Spectra from Each Probe
- Locate Synchronous Spectral Lines
- Read their Amplitudes
- Find the Largest
- Determine its Speed
- Retrieve Limits for L1(N1) Band
- Compare Speed With Limits
- Determine if Prove is V3
- Return Answers

Each of the Above Represents, In Turn, a String of Commands.

Figure 2-2

FOR SYSTEM AUTOMATION,
EACH STATEMENT IS IN FACT A
SERIES OF INSTRUCTIONS AND QUERIES

Implementation of each of the above steps requires, in turn, a separate string of commands in order to tell the system precisely what to do.

Each individual operation must therefore be expressed as a series of rules which are called by other rules leading to a highly complex system. Alternatively, a hierarchy of rule bases can be set up, each of which has a different function, such as diagnosis or data acquisition. The process is quite complex and difficult to visualize but does have the advantage that the rules can be combined in different orders by the inference engine to evaluate data in different ways.

Implementation of diagnostics using a fault tree approach also involves some twists which are not at first obvious. For example, a classical fault tree implementation assumes a "fault farm" consisting of individual faults which are observed by the system operator. An example of this is shown if Figure 2-3. A system fault or problem is detected by an operator whereby the diagnostic system proceeds to the appropriate tree, asks questions and makes decisions which eventually lead to a diagnosis. However, for a nonconsultative expert diagnostic system, the pre-existence of a fault cannot be assumed, much less its specific manifestation. Further, many faults may be present. Nonetheless, the basic fault tree approach is still valid when correctly applied.

The key to its application lies in getting the system to first find the information needed for a particular decision and then having the system use that information to make the decision. For example, Figure 2-4 shows the implementation of the same unbalanced low rotor turbine diagnostic logic which was shown in rule form in Figure 2-1. The blocks in Figure 2-4 instruct the system to search for certain values while the diamonds represent decision points. These blocks or "limbs" can be largely standardized with this approach in the form of, for example, operator queries, logical decisions, table look-ups or equation evaluations. Rather than a collection of diverse rules, the knowledge base becomes largely a collection of simple, standard building blocks or limbs to represent prespecified actions or user specified sequences within the limits of that limb's capability. The size of the fault tree can be kept within reasonable bounds by creating a number of individual fault trees, each representing a small subset of the overall number of faults. Since the blocks are standardized and recognized by the inference engine, their behavior is predictable. Therefore, there is less need of a knowledge engineer who becomes more of a guide to effi-

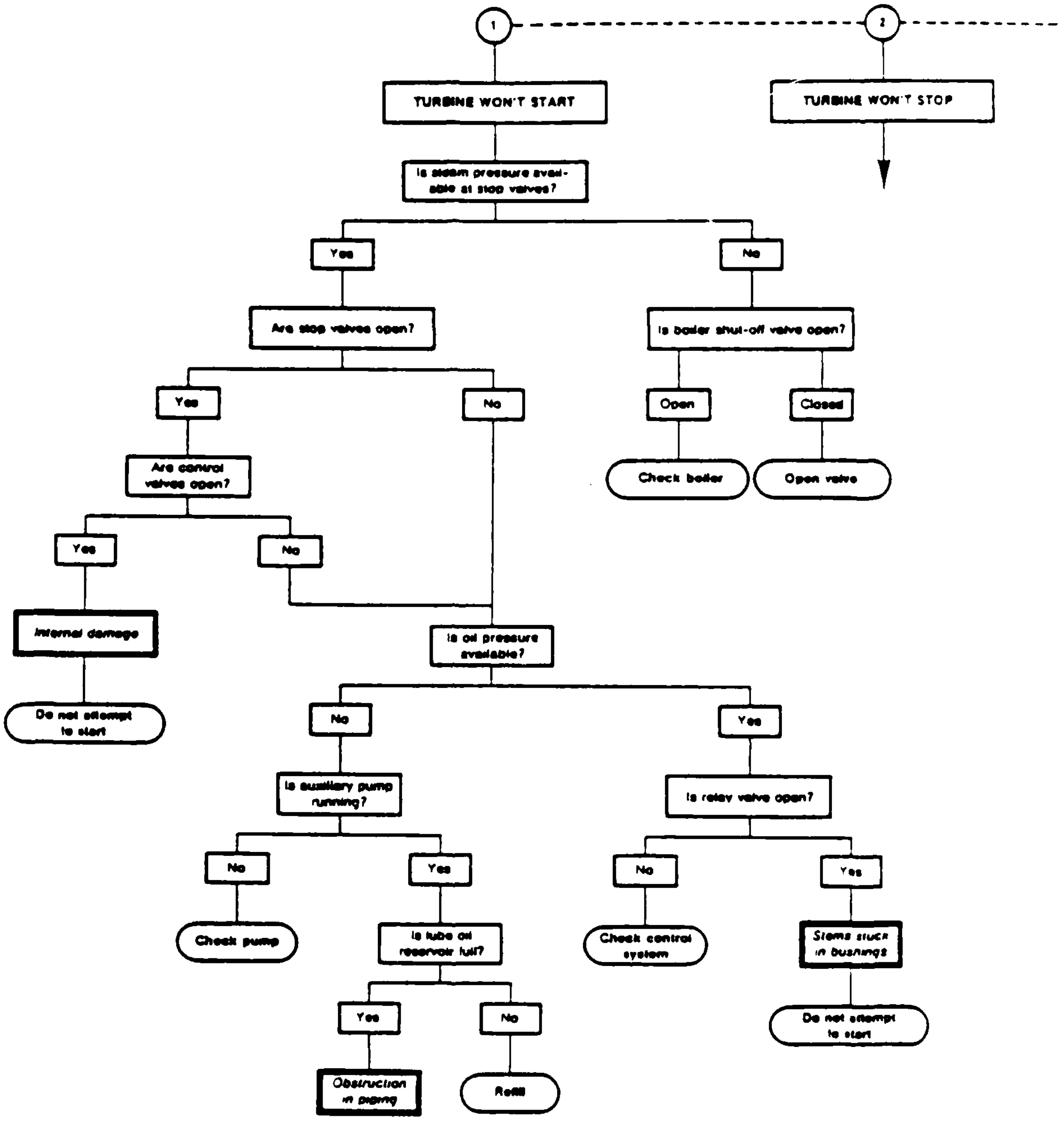


Figure 2-3

Example of Primitive Fault Tree
(Steam Turbine Troubleshooting)

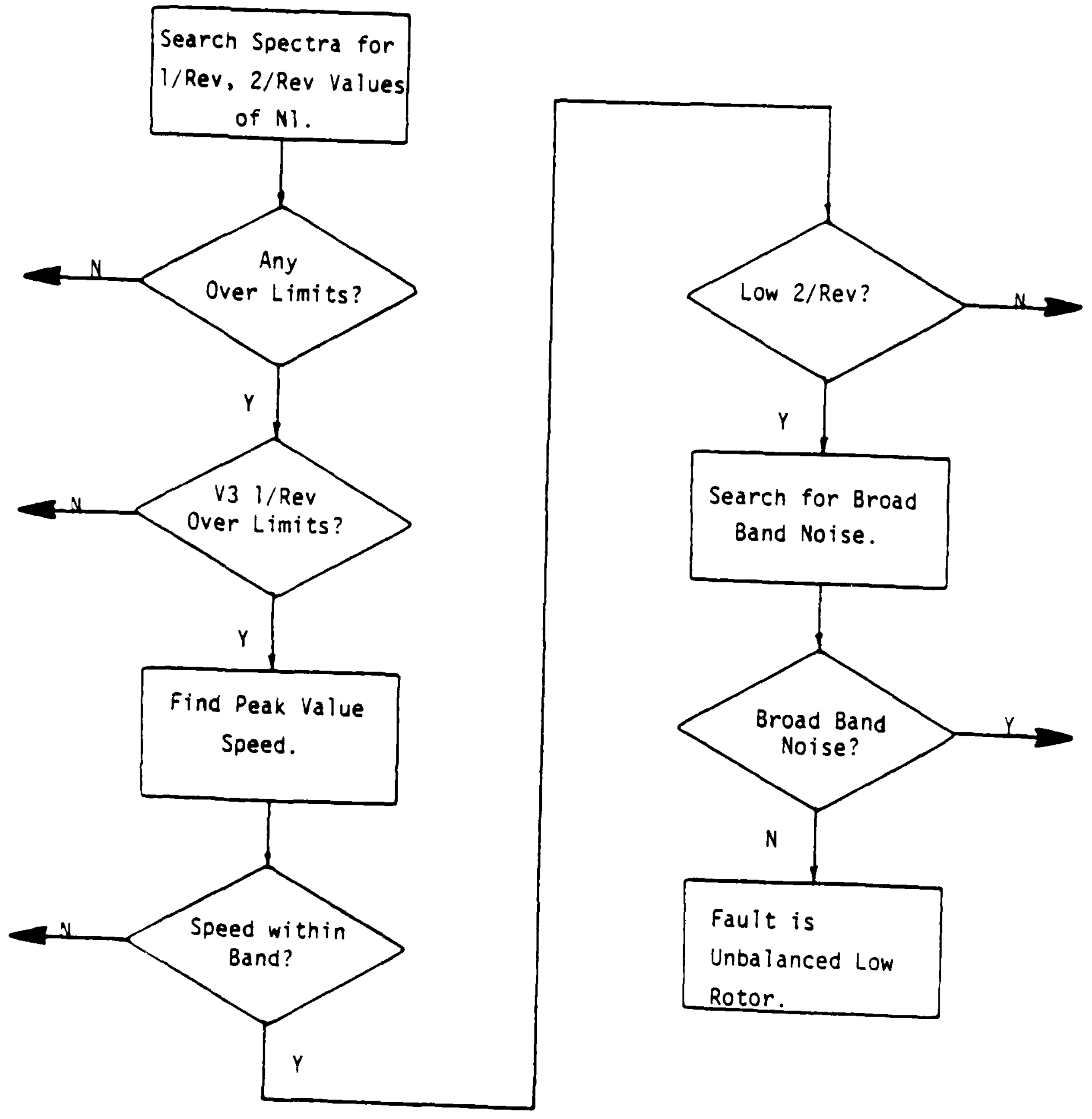


Figure 2-4

Example of Similar Diagnostic Scheme for Fault Tree System

cient representation of the knowledge rather than the central figure in development of the expert system.

The choice of a fault tree, as opposed to rule based, approach to diagnostics was made by MTI after an initial evaluation of both approaches. In particular, it was found that interaction with experts was greatly simplified when fault trees were used for the kinds of diagnostic problems which it was anticipated were to be addressed. In particular, reasons for selection for the fault tree approach included:

- Adaptability to the problem (experts often represent knowledge in terms of fault trees)
- Ease of visualization
- Minimization of contradictory rules
- Minimization of endless loops
- Ease of set-up for trees
- Simplification of inference engine design

This is not to say that an efficient rule based approach cannot be found, just that the fault tree approach currently appears to be the easiest, fastest path to develop a workable set of diagnostics for gas turbine engines.

In order to be easily implemented, fault tree based expert systems must have a series of standardized limbs or blocks which can be applied in various combinations to implement the diagnostics. Examples of the various types of limbs needed for this implementation are given below for three types of systems:

- A basic consultative system
- A nonconsultative system based on current needs
- A nonconsultative system based on anticipated future needs

The basic consultative system is one which gets its information from the system operator, that is, it asks a series of questions of the operator and uses the subsequent answers to arrive at a diagnosis. The system interacts with the user only and gives advice and may show pictures or provide instructions in acquiring test data or performing repairs. Typical limbs for this system would be:

- Start

- Query (decision based on question/answer)
- Advice/output, etc.
- Graphics output
- End

For the nonconsultative system currently being developed, all of the above limbs are necessary along with a number of new types. Since the system takes data automatically, reduces the data and operates on it, it requires a way to access these data and to make decisions. In addition, the large number of logical decisions which must be made would require an enormous, cumbersome fault tree. Therefore, a way to access multiple fault trees is needed. A way to make computations and evaluate equations would be useful as would a "special" limb for unusual requirements. Limbs to support these needs would be:

- Data acquisition
- Table storage/look up
- Logic
- Numerical input/output
- Calculation (equation evaluation)
- Tree jumps
- Special (user defined)

There are a number of needs for larger scale future systems. Most of these needs are in the user interface area and are directed towards making the system easier to set up and maintain and improve interpretation of output. In addition, it would be advantageous for the system to be able to control certain parameters. For example, the system could make use of control of vibration probe sensitivity ranges or FFT analysis options of a given portion of test data. Control of a machine or process to achieve certain values or test points is also not out of the question. To summarize, these future needs involve:

- Plotting capability
- Simplified user interface
- Knowledge based editing software
- Knowledge based visualization software
- Control capability
  - Instrumentation

  - Process

In addition, for all levels of diagnosis, techniques must be worked out within the fault tree structure to handle data which are unclear and to estimate the likelihood of a given fault being present.

2.6 DIAGNOSTIC STRATEGY FOR A REPRESENTATIVE SET OF ROTOR SYSTEM FAULTS

The purpose of this subsection is to illustrate the construction of a sample fault tree that could be implemented in a demonstration diagnostic system. The overall construction of the fault tree will be illustrated and one approach for increasing the efficiency of diagnostics and handling multiple faults will be described. In addition, it is intended that this fault tree be comprehensive enough to be used as a basis for the diagnostics implemented in the implant fault rig demonstration system, covered under Task 4 of this program. The overall strategy is to use the AVID™ system diagnostics as a basis, but simplified for the diagnosis of a system with a single rotating element. Therefore, emphasis can be placed on diagnosing individual rotor bearing problems without the complications of multiple rotors and multiple critical speeds implied in the original diagnostic set.

The specific faults to be diagnosed are:

- Rotor shift
  - One time
  - Recurring
- Unbalance
  - Small to moderate
  - Large, bowed rotor
- Misalignment
- Unbalance combined with misalignment
- Increased or decreased support flexibility
- Rubs/bearing faults

These diagnostics are shown in detail in Table 2-3. The table lists the individual faults, the symptoms associated with those faults, the information required for diagnosis and the action to be taken by the system. In setting up this diagnostic approach a number of assumptions have been made to provide a framework within which to simulate a "real world" application. These assumptions are that a single rotor is being diagnosed and that rotor has one bending critical speed in its operating range. Data from two sensors are assumed to be available. Rotor motion at low frequencies would be detected by a displacement sensor while high frequency casing motion would be seen by an accelerometer. It is further assumed that only data acquisition will occur during the operation of the rotor bearing system. Data will be stored and interpreted afterward by the expert system. Note that this approach is compatible with the way in which most modern machinery monitoring systems operate, that is spectra are typically acquired, stored and processed at some time following the data acquisition. In this case, it would be akin to simulating the monitoring of a gas turbine engine in a test cell following overhaul during an acceptance test. The function of the diagnostic system would be to acquire data during engine operation, reduce the data, interpret it, either "pass" or "fail" the engine and, if the engine fails due to high vibration, to perform a diagnosis which could isolate the cause or causes of the high vibration.

It is further assumed that sensors are operating properly and are delivering viable, reliable data to the system. In addition, because data are processed off-line, any checks of whether high vibration alarm limits have been exceeded are assumed to have been made separately.

Within this framework the overall approach to diagnostics would assume the following procedure:

1. The rotor is set to its initial speed.

2. Low and high frequency spectra are required from vibration probes at that speed.

3. The procedure is repeated for all remaining rotor speeds.

Table 2-3

REPRESENTATIVE SET OF ROTOR SYSTEM FAULTS

FAULT	SYMPTOMS	INFORMATION	ACTION
Rotor Shift (One Time)	Sharp Amplitude Change (1/rev on run-up) No change on run-down or subsequent runs.	1/rev amplitude vs. speed	Identify condition. Determine if blanacing necessary and feasible. If yes - balance rotor. If no - reject rotor.
Rotor Shift (recurring)	Sharp amplitude change (1/rev) on run-up, run-down Sharp changes on subsequent runs.	1/rev amplitude vs. speed	Identify condition. Reject Rotor.
Unbalance	Large synchronous amplitude at critical speed. Low 2,3/rev values. Balance correction within weight capacity.	1/rev amplitude at peak	Identify condition. Determine if balancing feasible. If yes-balance rotor. If no-reject rotor
Large Unbalance/ Bowed Rotor	Large synchronous amplitude at critical speed. Low 2,3/rev values. Balance correction 2-3 X weight capacity.	1/rev amplitude at peak	Identify condition. Reject rotor.
Misalignment	Large 2/rev amplitude. Low 3/rev amplitude. No broad band noise. 1/rev amplitude 2/rev.	2/rev amplitude at critical speed. 3/rev amplitude at critical speed. Broad band noise determination. 1/rev amplitude at ciritcal speed.	Identify. Reject rotor for reassembly.
Unbalance and Misalignment	Large 1/rev amplitude at critical speed. Large 2/rev amplitude at critical speed. Low 3/rev amplitude. No broad band noise. 1/rev amplitude substantially larger than w/rev.	1/rev amplitude at critical speed. 2/rev amplitude at critical speed. 3/rev amplitude at critical speed. Broad band noise determination.	Identify. Reject Rotor for reassembly/rebalance.
Increased/Decreased Flexibility	Large 1/rev amplitude at critical speed. Out-of-tolerance location of critical speed peak.	1/rev amplitude at critical speed. Frequency at critical speed.	Identify. Reject rotor for cracks/loose supports check.
Rubs/Bearing Faults	Large numbers of low/high frequency peaks or Broad Band noise at high frequencies or Large 1/rev, 2/rev, 3/rev amplitude.	Existence of multiple peaks. Existence of broad band noise. 1/ 3, 3/rev vs. speed.	Identify. Reject rotor for bearing/seal inspection.

4. Spectral data are reduced to an amount manageable by the diagnostic system. For example, if two 400 line spectra are required at each of ten speeds, a total of over 8000 data points must be evaluated. This potential problem will be circumvented by reducing data to a series of tables or lists containing only relevant data. The diagnostic system "knows" the format and general content of these lists and obtains its diagnostic information from them.

5. Lists of reduced data are passed to the diagnostic system software.

6. The diagnostic system performs the minimum amount of analysis at each diagnostic step which will enable it to make a decision and proceed to the next branch point. This minimizes the amount of searching and data interpretation which must be done at each branch point (no exhaustive searching) and simulates the approach of an expert. It also allows for accomodation of multiple faults.

7. A "passout" run will be simulated. That is, if all vibration levels are low, the system is in good condition and detailed diagnostics are unnecessary. If vibration levels are significant, the rotor is failed and the diagnostic system then attempts to seek out what is wrong.

In order to implement the diagnostics shown in Table 2-3 within the framework of procedures and assumptions listed above, an overall strategy is needed. This strategy is shown in Figure 2-5 which shows the overall diagnostic strategy in block diagram form. Going through the figure, we find that data are required as the rotor is brought to different speeds and data reduction is implemented to reduce these data to a series of lists or tables. The data from the low frequency sensor is then scanned for high vibration levels. If none are over limit, high frequency data are scanned and, if none exceed limits, the rotor is passed. Fault diagnosis occurs when either low or high frequency limits, or both, are exceeded.

This strategy is outlined in greater detail in the next five figures. Each of these figures represents an individual fault tree in the overall data acquisition and diagnostics process. The first tree, Data Acquisition-Limit Search, is shown in Figure 2-6. This is basically an expansion of Figure 2-5. However the

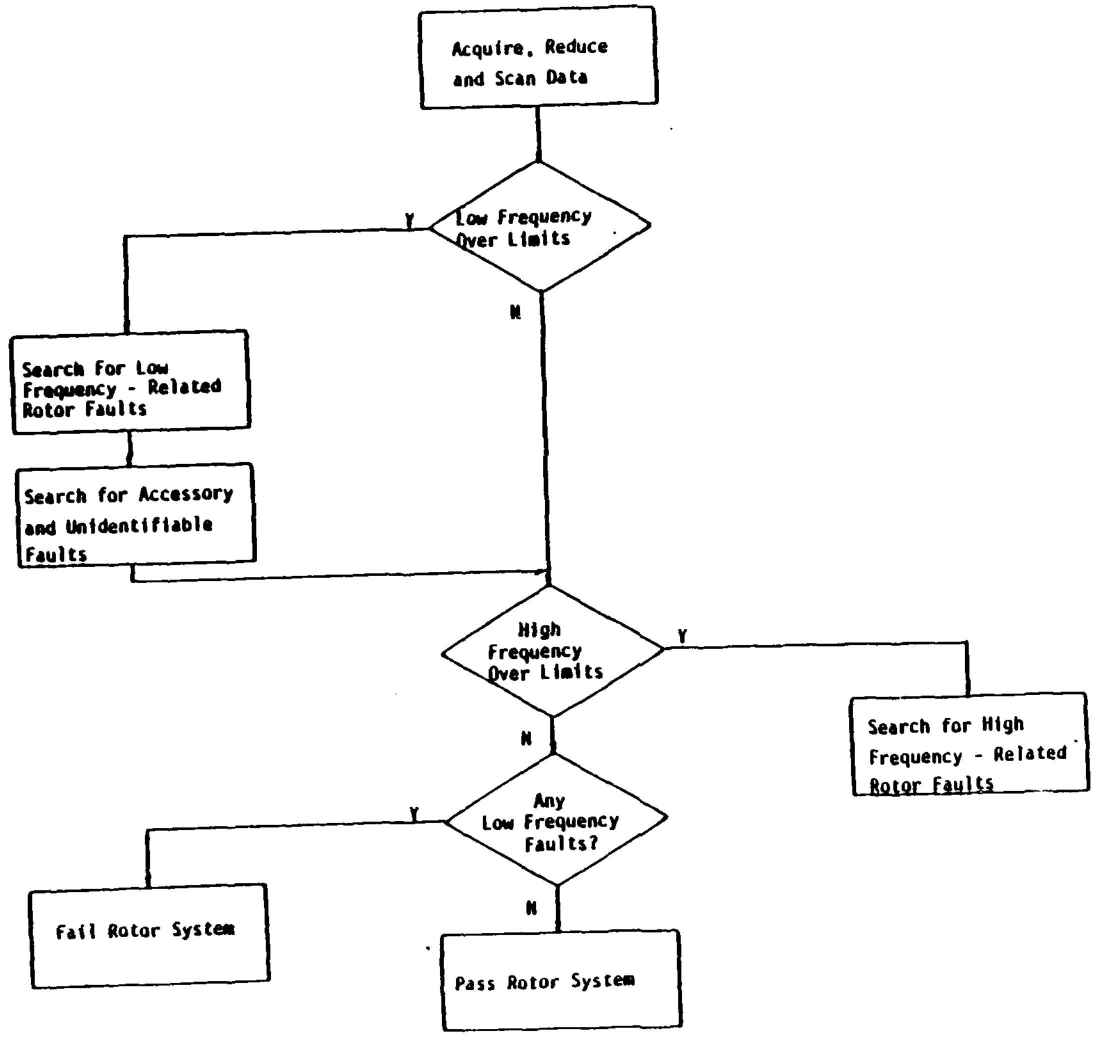

Acquire, Reduce and Scan Data
Low Frequency Over Limits
Y
N
Search For Low Frequency - Related Rotor Faults
Search for Accessory and Unidentifiable Faults
High Frequency Over Limits
Y
N
Search for High Frequency - Related Rotor Faults
Any Low Frequency Faults?
Y
N
Fail Rotor System
Pass Rotor System

Figure 2-5

Overall Diagnostic Strategy

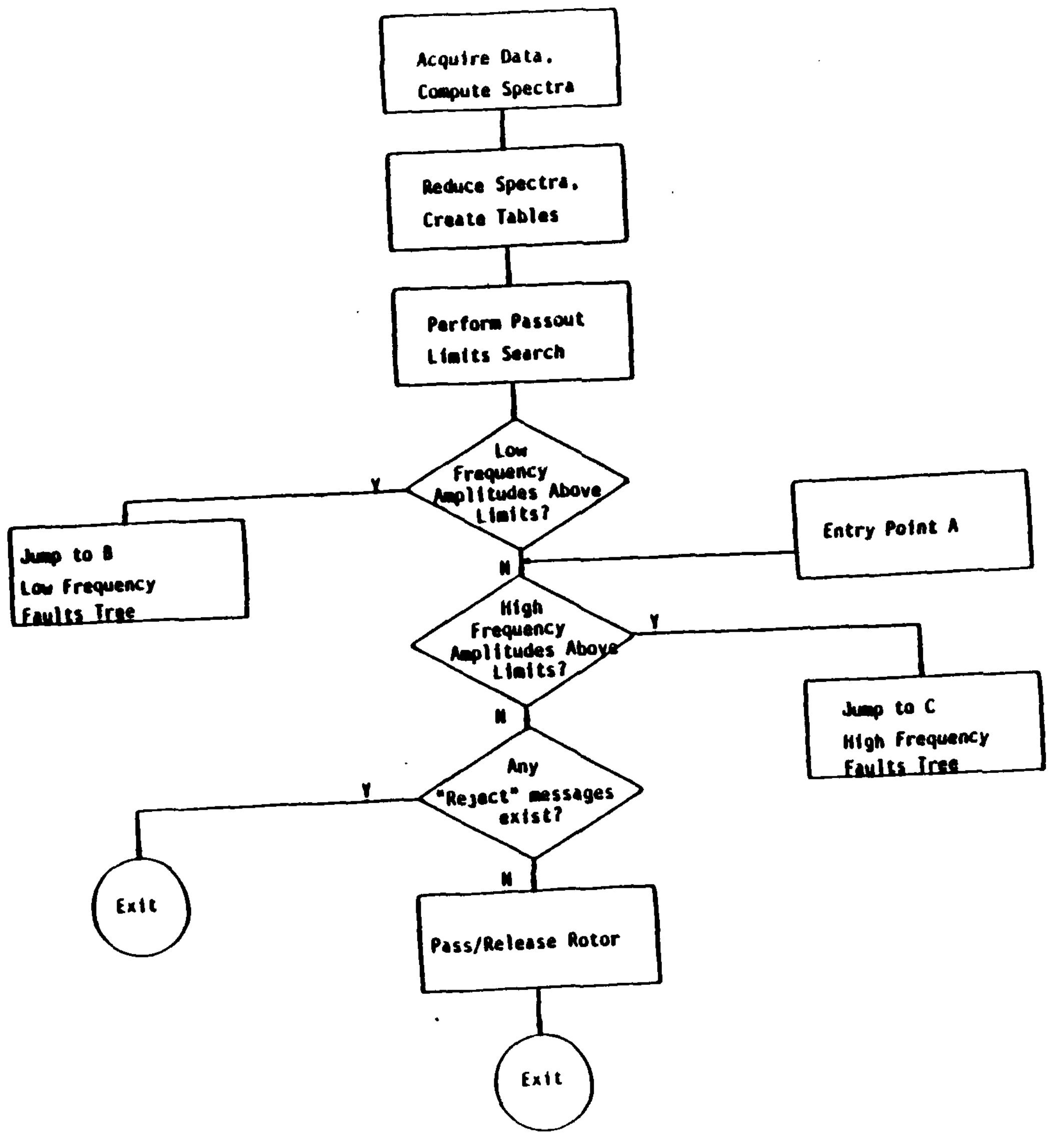

Acquire Data, Compute Spectra
Reduce Spectra, Create Tables
Perform Passout Limits Search
Low Frequency Amplitudes Above Limits?
Y
Jump to B Low Frequency Faults Tree
N
Entry Point A
High Frequency Amplitudes Above Limits?
Y
Jump to C High Frequency Faults Tree
N
Any "Reject" messages exist?
Y
Exit
N
Pass/Release Rotor
Exit

Figure 2-6

Data Acquisition - Limit Search Tree

limbs, which in the earlier figure simply describe fault searches, are now replaced with instructions to jump to the appropriate fault tree or to enter from a different tree.

Figure 2-7 describes the first of two trees that deal with the diagnosis of faults found through analysis of low frequency data. The strategy within this tree is to make a series of very simple checks, each of which is the answer to a specific question. The question is then asked by means of a logic limb and branching occurs depending on the answer. The way this specific diagnostic algorithm is laid out, the initial check is made for high synchronous rotor amplitudes. If a large one per/rev component of vibration amplitude is found, faults associated with synchronous response are checked for. If large amplitudes are not found, a different series of faults is sought. Given a large synchronous amplitude, the first thing that is checked for is loose or shifting parts in the rotor. If loose parts do exist, the rotor is failed and all other one per/rev diagnostics are irrelevant as it will be impossible to separate unbalance or other related faults from the effect of the shifting unbalance in the rotor itself. Therefore, if this happens, further one per/rev diagnostics are sidestepped and the system begins to check for faults related to other frequencies.

If no evidence of loose parts exists, the system then checks if the critical speed occurs in the expected region. Higher or lower than expected locations for this resonant peak could indicate, for example, a locked-up damper or cracked supports. Presence of either of these problems is cause to reject the rotor system and jump to that section of the fault tree which looks for misalignment or rub related problems.

If critical speeds are within range, a jump occurs to the second tree for low frequency faults shown in Figure 2-8. In this tree, the object is to diagnose unbalance, misalignment or a combination of the two as well as the possibility of a rub or a very serious fault in the rolling element bearings. If unbalance appears to be the problem, the system checks to determine whether in fact the rotor can be balanced. For example, the weight may be too large or may be required to be placed in an inaccessible location. If the rotor can be balanced, then balancing would be recommended. If the evidence points toward misalignment, misalignment combined with unbalance or rubs and bearing faults,

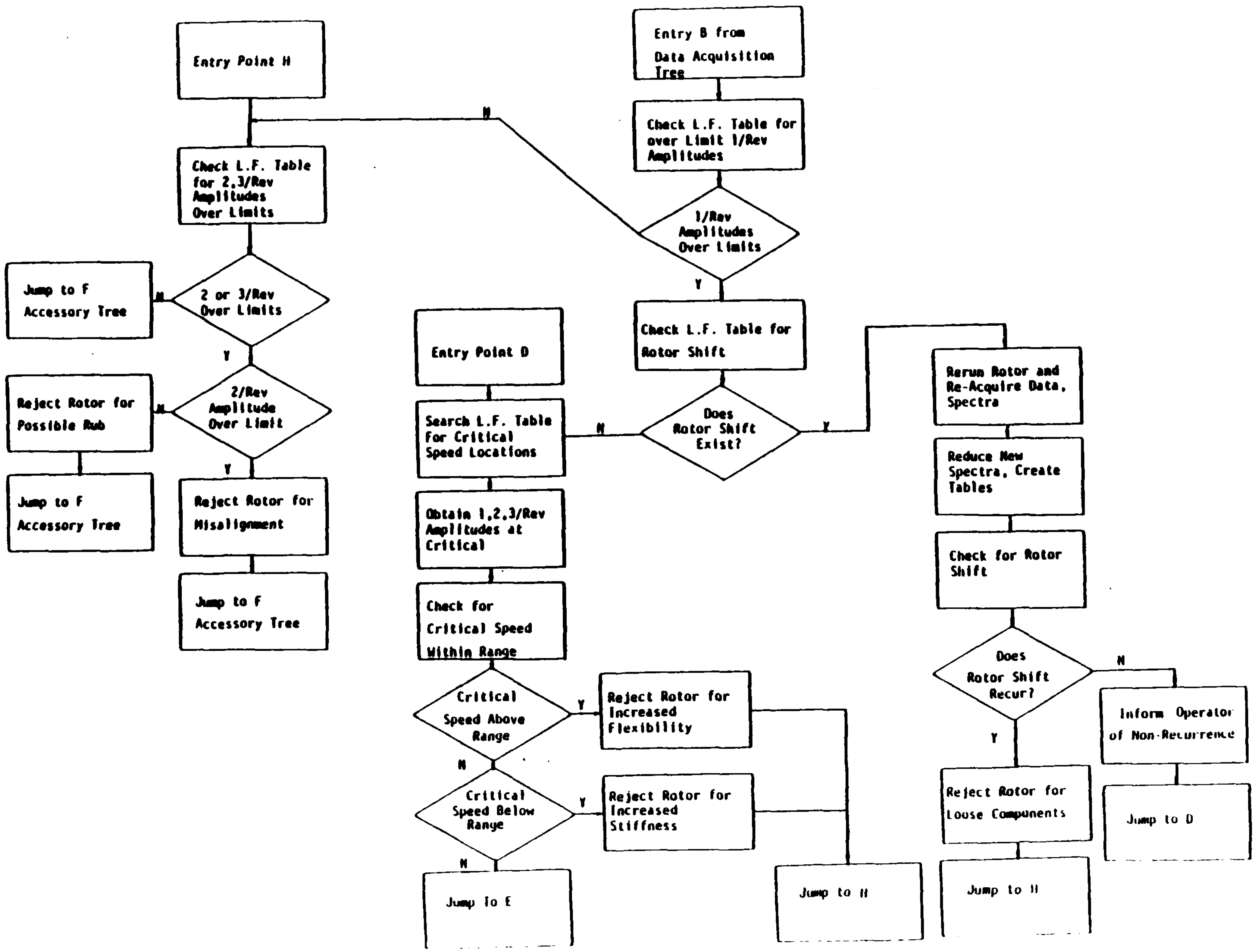
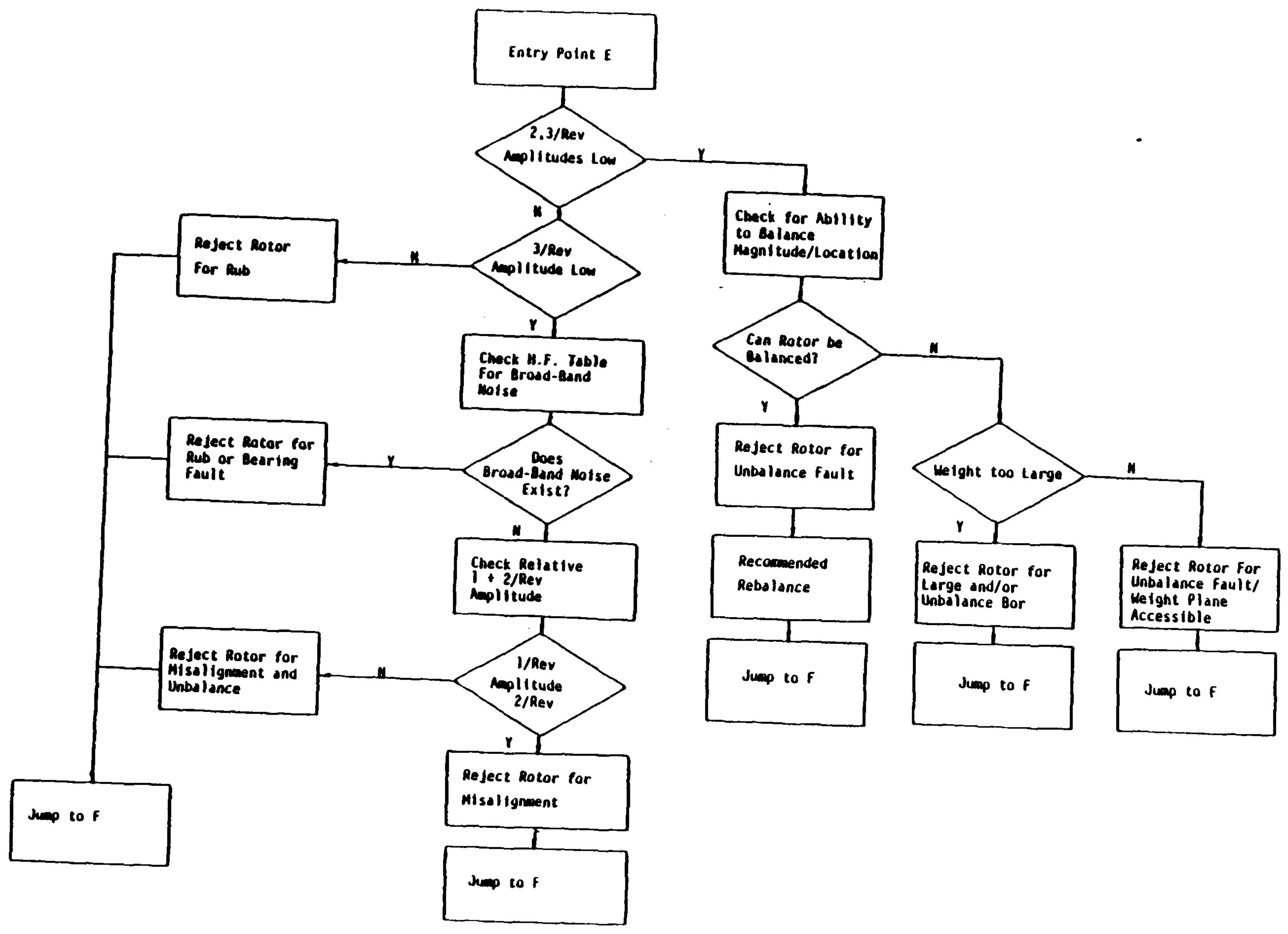

Entry Point H
Check L.F. Table for 2,3/Rev Amplitudes Over Limits
2 or 3/Rev Over Limits
N
Jump to F Accessory Tree
Y
2/Rev Amplitude Over Limit
N
Reject Rotor for Possible Rub
Jump to F Accessory Tree
Y
Reject Rotor for Misalignment
Jump to F Accessory Tree
Entry B from Data Acquisition Tree
Check L.F. Table for over Limit 1/Rev Amplitudes
1/Rev Amplitudes Over Limits
N
Y
Check L.F. Table for Rotor Shift
Does Rotor Shift Exist?
N
Entry Point D
Search L.F. Table For Critical Speed Locations
Obtain 1,2,3/Rev Amplitudes at Critical
Check for Critical Speed Within Range
Critical Speed Above Range
Y
Reject Rotor for Increased Flexibility
N
Critical Speed Below Range
Y
Reject Rotor for Increased Stiffness
N
Jump to E
Jump to H
Y
Rerun Rotor and Re-Acquire Data, Spectra
Reduce New Spectra, Create Tables
Check for Rotor Shift
Does Rotor Shift Recur?
Y
Reject Rotor for Loose Components
Jump to H
N
Inform Operator of Non-Recurrence
Jump to D

Figure 2-7 First Low Frequency Fault Tree



Entry Point E
2,3/Rev Amplitudes Low
Y
N
Check for Ability to Balance Magnitude/Location
3/Rev Amplitude Low
N
Y
Reject Rotor For Rub
Can Rotor be Balanced?
N
Y
Check H.F. Table For Broad-Band Noise
Does Broad-Band Noise Exist?
Y
N
Reject Rotor for Rub or Bearing Fault
Reject Rotor for Unbalance Fault
Weight too Large
N
Y
Check Relative 1 + 2/Rev Amplitude
Recommended Rebalance
Reject Rotor for Large and/or Unbalance Bor
Reject Rotor For Unbalance Fault/ Weight Plane Accessible
1/Rev Amplitude 2/Rev
N
Y
Reject Rotor for Misalignment and Unbalance
Jump to F
Jump to F
Jump to F
Jump to F
Reject Rotor for Misalignment
Jump to F

Figure 2-8

Second Low Frequency Fault Tree

these conditions are identified and, in any case, the system is instructed to jump to the next tree described in Figure 2-9.

This tree, which looks for accessory and unknown faults, can be entered in two ways. Either vibration limits have been exceeded but are not involved in the one, two or three per/rev components searched for in the previous two trees, or it is entered automatically after faults have been diagnosed involving those frequency components. In this latter case the system is looking for additional faults which, as can be seen, are easily accommodated by jumping to different tree limbs or trees once one problem has been diagnosed.

The Accessory and Unknown Fault tree first looks for vibration at frequencies normally associated with accessories which are usually gear driven and, therefore, run at some multiple of shaft speed. If high vibration is found at these frequency components, the system identifies the "problem" accessory and then searches the low frequency table for amplitudes at any frequencies which have not been previously defined. For example, with this diagnostic logic, a rotor may be undergoing sub-synchronous vibration. This could be vibration at a natural frequency corresponding to an already traversed critical speed due to, for example, excitation forces at blade tip clearance regions. This type of vibration, unstable and therefore highly dangerous, would be picked up as a large amplitude component in the frequency spectrum which does not correspond to a known fault. In this case the diagnostic system rejects the rotor and identifies to the operator the relevant data on speed, frequency and amplitude for a more detailed diagnosis to be made outside of its capabilities. Note that it is vital to have a diagnostic system which is capable of recognizing that rotor system problems may exist which it cannot understand. It must have a way of identifying this fact, of alerting the operator and giving that person the necessary information so that diagnostics external to the system may be performed.

Once the diagnostic algorithm has gone through this tree, control is returned to the data acquisition tree (Figure 2-6) and if large amplitudes in the high frequency region from the accelerometer exist, the High Frequency Faults tree is entered. This is shown in Figure 2-10. This tree looks for specific characteristics to define a rub or identify a very gross bearing fault, identifies these problems and terminates execution of the diagnostic algorithm. It also checks

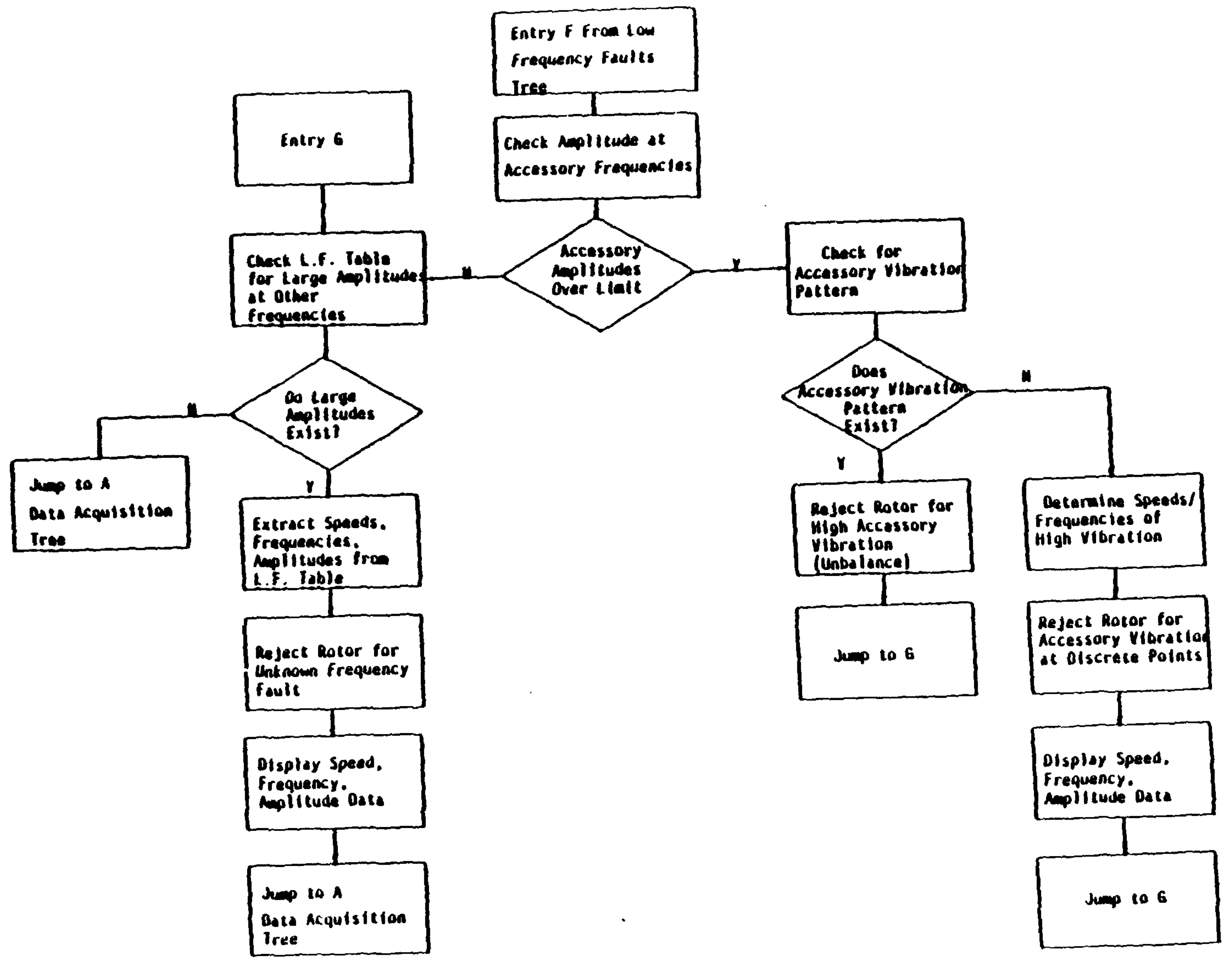

Entry F From Low Frequency Faults Tree
Check Amplitude at Accessory Frequencies
Accessory Amplitudes Over Limit
Y
Check for Accessory Vibration Pattern
Does Accessory Vibration Pattern Exist?
Y
Reject Rotor for High Accessory Vibration (Unbalance)
Jump to G
N
Determine Speeds/ Frequencies of High Vibration
Reject Rotor for Accessory Vibration at Discrete Points
Display Speed, Frequency, Amplitude Data
Jump to G
N
Entry G
Check L.F. Table for Large Amplitudes at Other Frequencies
Do Large Amplitudes Exist?
N
Jump to A Data Acquisition Tree
Y
Extract Speeds, Frequencies, Amplitudes from L.F. Table
Reject Rotor for Unknown Frequency Fault
Display Speed, Frequency, Amplitude Data
Jump to A Data Acquisition Tree

Figure 2-9

Accessory and Unknown Faults Tree

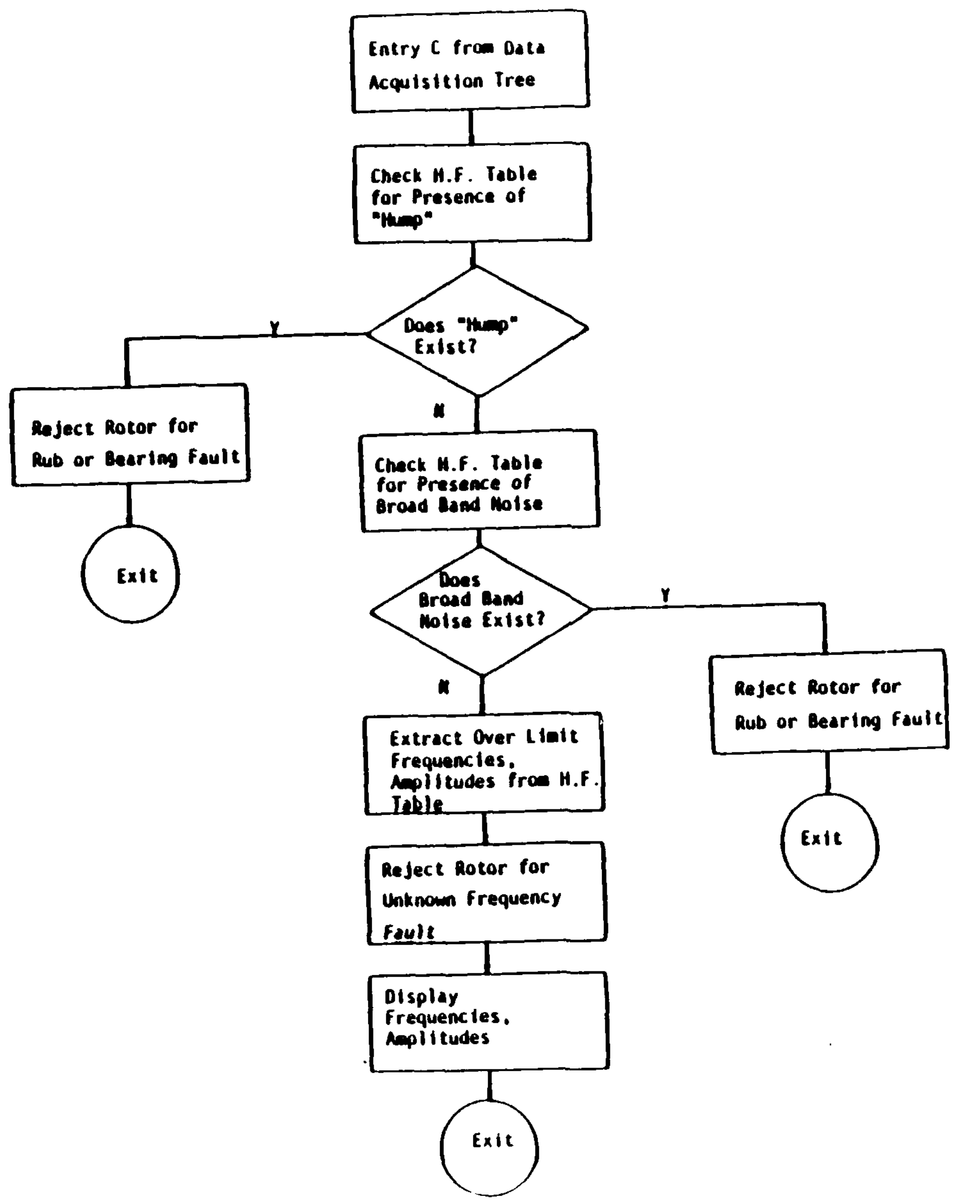

Entry C from Data Acquisition Tree
Check H.F. Table for Presence of "Hump"
Does "Hump" Exist?
Y
N
Reject Rotor for Rub or Bearing Fault
Exit
Check H.F. Table for Presence of Broad Band Noise
Does Broad Band Noise Exist?
Y
N
Reject Rotor for Rub or Bearing Fault
Exit
Extract Over Limit Frequencies, Amplitudes from H.F. Table
Reject Rotor for Unknown Frequency Fault
Display Frequencies, Amplitudes
Exit

Figure 2-10 High Frequency Faults Tree

for the existence of vibration related to faults which it is unable to identify. It alerts the operator of this fact and exits.

The rotor system diagnostic fault tree described above is of course not unique and can be further developed to increase its efficiency and potential effectiveness. However, it does describe a way to search for a number of rotor bearing system related faults, to do a minimal amount of searching to identify the presence of a fault before diagnosis is attempted and to diagnose multiple faults. It is also relatively straightforward to expand. Such expansions might involve accommodating diagnostics for multiple rotor systems or incorporating much more precise diagnostics of bearing faults. A further possible expansion of the system is to equip it with inputs describing the various system performance parameters and include trees which have calculation limbs that are able to compute actual performance and compare it with expectations. Diagnosis of performance and vibration problems in an integrated system would then be a reality.

To conclude, it is apparent from the above work that specific diagnostics can be developed for rotordynamic faults typically seen in gas turbine engines. These diagnostics can be implemented using fault tree logic and arranged in such a way as to keep system complexity at a reasonable level and accommodate multiple faults. Finally, additional faults can be incorporated either by adding additional limbs to the existing fault trees or by incorporating new fault trees to the knowledge base.

2.7 IMPLEMENTATION ISSUES

From the standpoint of rotor diagnostics considerations, a number of points need to be addressed to develop a viable, expert diagnostic system which is nonconsultative. This would involve expansion of the diagnostic system beyond the demonstration stage, and closer to one which could be implemented for diagnosis of gas turbine problems. The items which must be addressed are, specifically:

- A multiplicity of
  - rotors
  - critical speeds
  - vibration probes

- Capability to accommodate engine runs which are interrupted, that is which are terminated due to high vibration or performance problems.
- The performance of on-line checks of sensor data to ensure that data are viable and safe amplitude limits are not exceeded.
- Development of a way to calculate the probability of existence for a given fault.
- Ways to handle unclear data.
- Detection of faults in sensors, wiring, electronics and the monitoring system computer itself.
- Expansion of diagnostics capabilities
  - more faults
  - more comprehensive diagnostics
- A focus on the needs of an individual application
  - individual requirements
  - integration with inspection/maintenance cycle

The details of exactly how far to expand the system in any given area must await the identification of exact location and application of the diagnostic system.

CHAPTER 3 ESTABLISHMENT OF EXPERT SYSTEMS FEATURES (TASK 3)

3.1 OVERVIEW OF EXPERT SYSTEMS

The objective of this task was to define the features required of an expert system for rotor diagnostics based on the diagnostic strategy developed in Task 2. The capabilities of existing expert systems and symbolic programming languages were identified in a comprehensive survey. The results from the survey were then used to define the features required of an expert rotor diagnostic system.

Conventional computer programs usually process numerical data, with the results also being numbers presented as lists, tables or graphs. It is left to the user to interpret the data. Novices generally do not have the knowledge needed to interpret what the numbers imply about the state of the object or system they refer to. In contrast, an expert uses a store of private knowledge consisting of facts, rules of thumb and problem solving methodologies derived from his experience to interpret the numbers and recommend solutions. An expert's private knowledge is generally heuristic knowledge which, if made available to non-experts, will enable them to perform at a higher level than before. Expert systems are computer programs that provide high levels of performance by combining the private heuristic knowledge of an expert with the public knowledge available on a specialized subject. An expert system uses facts provided by the user, the knowledge stored in the system and general problem solving procedures to solve a particular problem.

An expert system has several components: a knowledge base, an inference engine, a user interface and an explanation facility. The knowledge base contains all the rules and facts about the particular subject, called the domain, of the expert system. The inference engine provides the problem solving procedures used in the expert system. The user interface allows the user to interact with the expert system to answer questions and provide facts related to the particular problem being solved. The explanation facility allows the user to query the expert system about its reasoning process by asking why a specific fact is required or how the system arrived at a specific conclusion. A schematic of an expert system is shown in Figure 3.1. The separation of knowledge from the mechanics of its use in solving a problem is a significant feature of expert

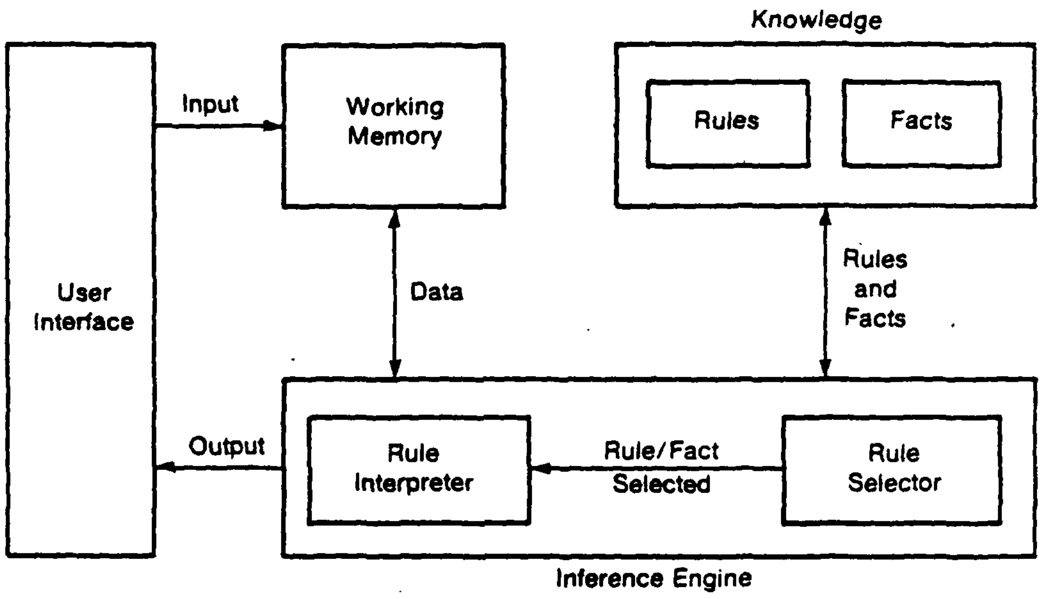


Figure 3-1

Architecture of a Simple Expert System

systems. By treating each rule as an independent unit of knowledge, expert systems permit an incremental development of the knowledge base by trial and error. The problem solving power of the expert system grows as more knowledge is added to the system.

Expert systems are beneficial when experts are in short supply relative to the demand for their services, the use of human experts is not cost effective, retention of corporate expertise is important in the event of loss of key personnel and when new personnel need to be trained to perform highly specilized functions unique to an organization. While an expert system cannot replace a human expert, it can be designed to handle the problems an expert considers routine and thereby free the expert to concentrate on unusual problems. The level of expertise of people performing the same function in a organization can vary, resulting in unacceptable variation in the quality of work as individuals improvise ways to solve problems. Expert systems can help improve the quality of work by standardizing work procedures.

Expert systems should be used with care. Reliance on deductive logic can result in erroneous conclusions because the rules of thumb used by an expert are not precise or certain. Suitable methodologies for ensuring the consistency and completeness of knowledge bases are not yet available and a novice user generally has no means of detecting a wrong answer. Explanations provided by current expert systems are superficial in that they refer the user to the rule being used but take the validity of the rule for granted. The inability to provide deeper explanation is inherent in the heuristic nature of an expert's knowledge. Expert systems that combine a deeper model-based knowledge about the domain with an expert's heuristic knowlege are now being developed to provide a more robust framework for solving problems.

Expert systems have been used successfully for diagnosis, planning, design, control and training aids. Some existing expert systems are reviewed in Appendix C.

3.1.1 Implementing an Expert System

Implementing an expert system involves the following steps: selecting a problem and identifying the knowledge to be included in the system, selecting an expert system development tool suitable for the domain, designing the system, developing the prototype to test the design and, finally, expanding and testing the system until it is complete. Knowledge base maintenance and update procedures should also be developed to ensure that future enhancements will be consistent with the design of the system.

The problem selected for the expert system should be a well structured problem where the procedures can defined clearly. For example, a diagnostic expert system will be of little value if the modes of failure in the machine being diagnosed change frequently in an unexpected manner. There are, however, no precise, reliable criteria currently available to decide which problems are solvable using expert systems. Identification of domain experts and extraction of their knowledge is the most important part of the development effort. Knowledge extraction requires a knowledge engineer familiar with the capabilities and limitations of the expert system development tool to interview the expert and transform the knowledge into a form suitable for the expert system. The knowledge engineer and the expert work together over the entire development cycle because it takes several iterations before the design of the expert system and the contents of the knowledge base can be finalized. The knowledge engineer must ensure that the rules representing an expert's knowledge are used in a way consistent with underlying assumptions.

A number of expert system development tools are now available. The tools usually include an inference engine to implement a specific problem solving technique such as forward or backward chaining, a suitable knowledge representation scheme and support programs such as knowledge base editors. If such tools are suitable for the problem domain, the development time can be reduced significantly. If no suitable commercial tools are available, a customized expert system has to be developed using programming languages such as C, LISP and PROLOG. Customized expert systems are, however, very expensive to develop.

3.2 FEATURES REQUIRED OF AN EXPERT SYSTEM FOR ROTOR DIAGNOSTICS

An expert system used to implement the diagnostic strategy developed in Task 2 should have the ability to accept vibrations data from a rotor, use a convenient knowledge representation scheme and be easy to use. These features are discussed below.

3.2.1 Data Acquisition System

Vibration data may be acquired by the expert system in one of two ways: a data acquisition function which is an integral part of the expert system or an independent data acquisition system which acquires data at the request of the expert system and transfers the data to the expert system. An integrated system will provide better performace but will be more difficult to develop, test and maintain. A modular system will allow the data acquisition module to use hardware and software optimized for its function without being constrained by the needs of the expert system as long as the interface requirements are met. Issues specific to data acquisition such as sensor failure, signal conditioning and data reduction can be dealt with separately from the issues relating to the use of the data by the expert system. While the performance of a modular data acquisition system will be lower than that of an integrated system, it will be easier to develop, test and maintain. The advantages of a modular system outweigh the incremental decrease in performance.

3.2.2 Knowledge Representation Scheme

It was established in Task 2 that fault trees are used by rotor dynamic diagnostic experts to express their knowledge. Fault trees are, however, static and difficult to maintain. Every path through the tree must be defined precisely. While an expert may be able to represent specific elements of a diagnostic strategy as a well defined fault tree, the overall diagnostic strategy incorporating those elements may not be defined precisely enough for a fault tree. A rule based expert system should, therefore, be used. The expert can specify individual rules while the inference engine is relied upon to invoke the rules as needed.

3.2.3 User Interface

The expert rotor diagnostic system must be easy to learn and easy to use. The terminology used to describe the technical procedures and the information display formats used should be suitable for the level of expertise of the intended users. Help should be available when needed by the user and the system must detect the more common user errors.

The degree of decision making delegated to the expert system versus that retained by the user is an important design decision. For example, a diagnostic system meant for use by novice users unfamiliar with the problem domain should minimize user data input which requires interpretation. For example, the user should be asked for a temperature reading rather than an assessment of the temperature being normal, high or low. The novice user should be guided through each step of performing a function but the number of steps should be kept small enough to avoid tedium. The results of the diagnosis should be presented in an unambigious manner consistent with the novice user's ability to understand the results and to detect a wrong diagnosis when it occurs. Contradistinctively, a diagnostic system for use by more experienced users should be advisory in nature, helping the user to arrive at a diagnosis but not itself making a final diagnosis.

3.3 CONFIGURATION FOR AN EXPERT SYSTEM FOR ROTOR DIAGNOSTICS

The recommended system configuration for an expert system based rotor diagnostic system is shown in Figure 3.2. The diagnostic system will have two modules: rotor diagnostics expert system and data acqusition.

The rotor diagnostics expert system will provide the user interface and incorporate the knowledge for fault detection, fault isolation, and recommended maintenance procedures. It will send a request for data to the data acqusition module when data are needed for diagnosis. The diagnostic expert system will also incorporate the ability to diagnose problems using alternate strategies if the data acquisition module is unable to provide the data needed for the current

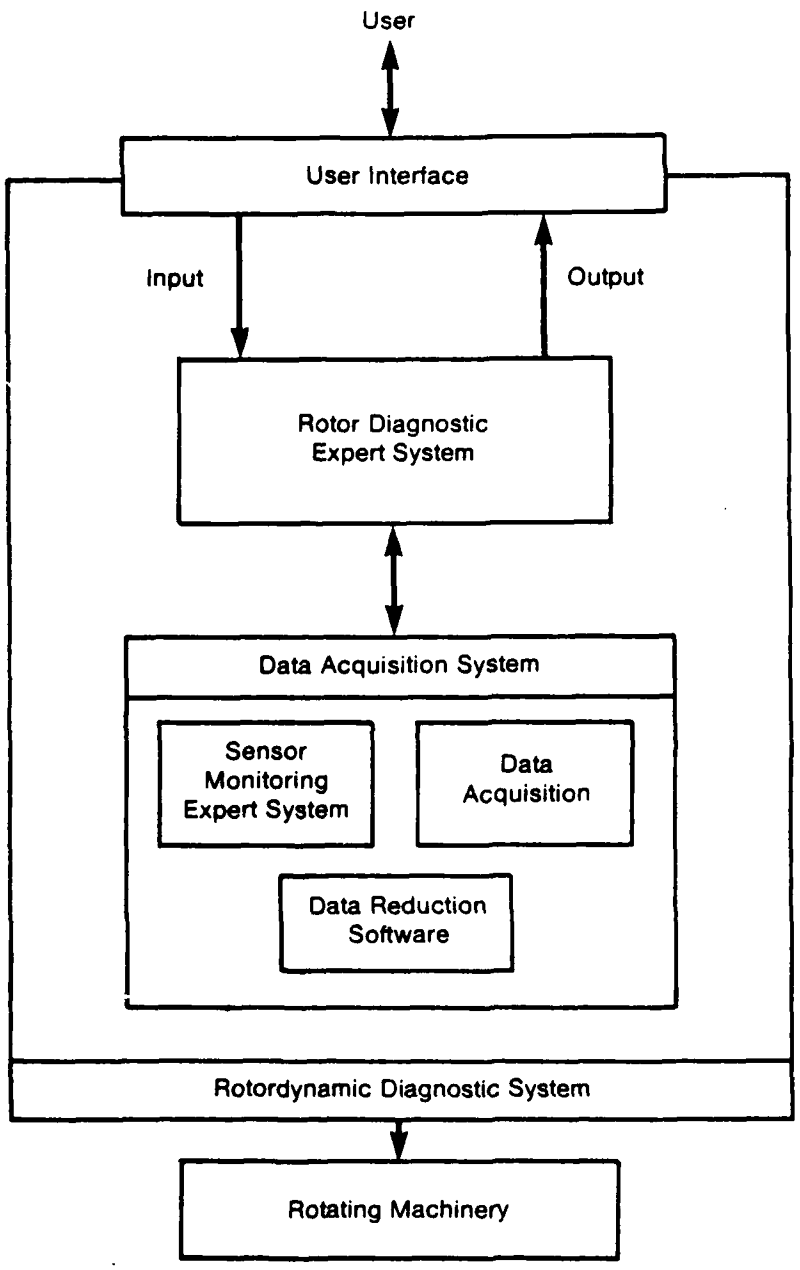


Figure 3-2

Rotor Diagnostic System Configuration

diagnostic strategy because of sensor failures. Maintenance personnel will be alerted to perform the necessary repairs.

The data acquisition module will be a self-contained module with the ability to exchange data with the diagnostic module. The module will process sensor signals and send the resulting data to the diagnostic module. Raw sensor data will not be sent. The data acquisition module will include an expert system to select the sensors to be used to acquire the data needed by the expert system. The sensor complement for the diagnostic system will be designed to provide alternate means to acquire the same data. In case of sensor failure, the sensor monitoring expert system will inform the diagnostic module of the failure and select an alternate set of sensors to be used to acquire data. If the data requested by the expert system cannot be acquired using the remaining sensors, the diagnostic module will be so informed and the data acquisition module will wait for the diagnostic module to select a new diagnostic strategy.

CHAPTER 4 DEMONSTRATION OF AI BASED DIAGNOSTIC SYSTEM CONCEPT (TASK4)

A demonstration system using existing MTI hardware and software resources was implemented to prove the feasibility of the diagnostic strategy developed for gas turbine engines. A single rotor test rig with the capability to accept implanted faults was selected and a diagnostic strategy tailored for that rig was implemented using a fault tree analyzer. A schematic of the demonstration system is shown in Figure 4.1.

4.1 ROTORDYNAMIC TEST RIG

A detailed description of the test rig is given in Appendix A. A schematic of the test rig is shown in Figure 4.2. The rotor consisted of an aluminum tube with a steel stub shaft on each end of the tube. A steel disc was mounted on each of the stub shafts. The rotor was supported by three deep groove ball bearings mounted in rigid bearing pedestals.

Five faults could be implanted in the test rig: unbalance, misalignment, increased bearing support flexibility (broken support), rub and accessory vibrations. Unbalance was implanted by adding weights of approximately 0.2 gm to one of the balancing planes (discs 2 and 3 in Figure 4.2). Misalignment was introduced by changing the radial position of bearing 2 relative to bearings 1 and 3. Increased bearing support flexibility was simulated by changing the radial stiffness of the bearing 2 pedestals from a baseline value of 35 MN/m to 1.75 MN/m. Rub fault was implanted by rubbing a stationary bolt against disc 4. A bolt was mounted with a clearance of 0.04 mm between the disc and the bolt. The rotor was then deliberately unbalanced such that the the resulting shaft deflection at disc 4 was sufficient for the bolt to rub against the disc. Accessory vibrations were simulated by connecting an electro-dynamic shaker to probe 4 bracket to introduce an external vibration with a frequency of 0.47 times the critical speed of the rotor system.

Four proximity probes were used to measure vibrations. The locations of the probes were selected to provide the best signals for the vibrations caused by each of the five faults implanted in the test rig. A fiber optic sensor was used

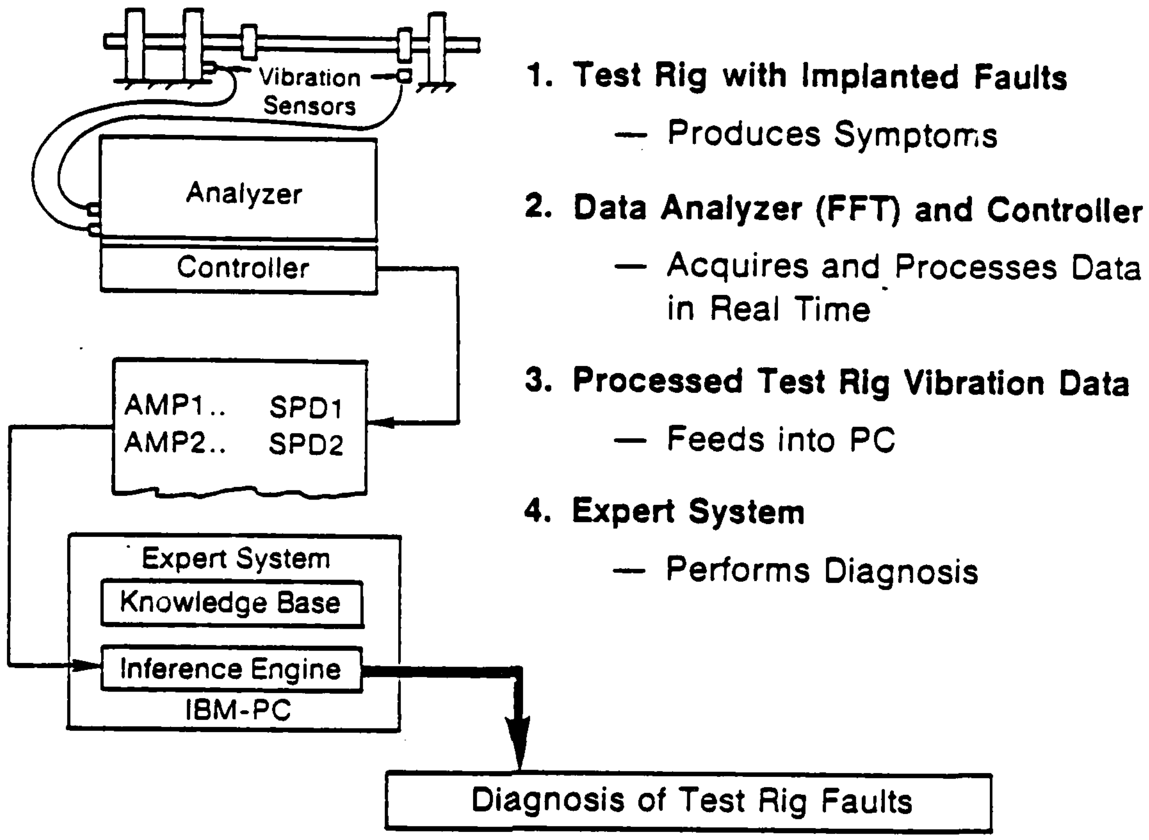
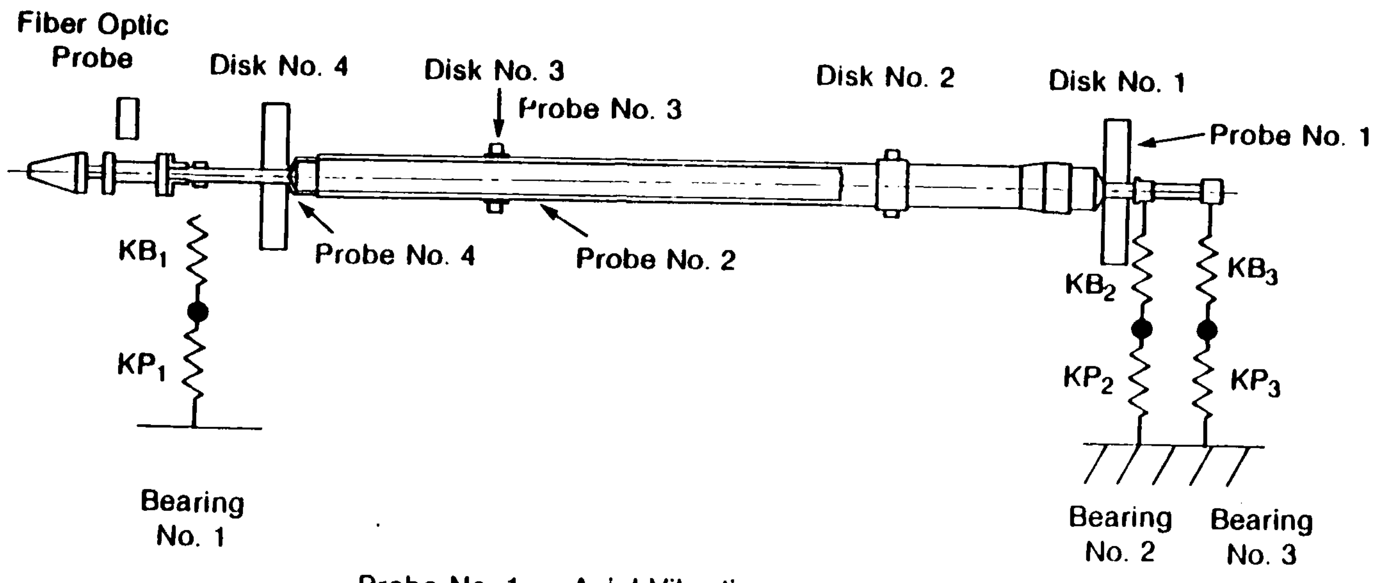

Vibration Sensors
Analyzer
Controller
AMP1.. SPD1
AMP2.. SPD2
Expert System
Knowledge Base
Inference Engine
IBM-PC
Diagnosis of Test Rig Faults
1. Test Rig with Implanted Faults
— Produces Symptoms
2. Data Analyzer (FFT) and Controller
— Acquires and Processes Data in Real Time
3. Processed Test Rig Vibration Data
— Feeds into PC
4. Expert System
— Performs Diagnosis

Figure 4-1

Rotordynamic Demonstration System



Probe No. 1 — Axial Vibrations
Probe No. 2 — Horizontal Radial Vibrations
Probe No. 3 — Vertical Radial Vibrations
Probe No. 4 — Horizontal Radial Vibrations

Figure 4-2

Schematic of Test Rig

86690

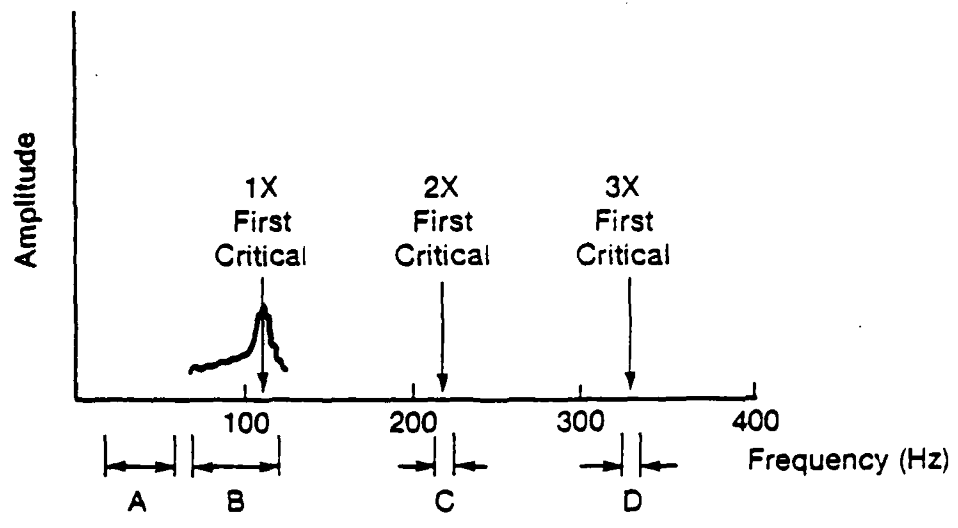
to measure rotor speed and phase reference. The locations of the probes are shown in Figure 4.2. Probe 1 was used to measure the axial response due to misalignment. Probes 2 and 3 were used to measure horizontal and vertical response, respectively, at the center of the rotor because of unbalance and increased support flexibility. Probe 4 was used to measure the response because of rub and accessory vibrations.

A mathematical model of the test rig was used to determine the critical speeds of the rig for the baseline case with no implanted faults. The first forward and retrograde critical speeds were calculated to be 6,546 and 5,463 r/min, respectively.

4.2 DATA ACQUISITION SYSTEM

The data acquisition system was designed to acquire data at the request of the diagnostic system, to process the data to derive the information needed by the diagnostic system and to send the information to the diagnostic system in the required format. The data acquisition system, described in Appendix A, consisted of a fast fourier transform (FFT) analyzer to convert the sensor signals into frequency domain data under the control of a microcomputer used for data collection and processing. Two channels, labeled A and B, were available for sensor signal input into the FFT analyzer. Channel A was used to monitor probe 2 while channel B was used to monitor either probe 1 (axial vibrations) or probe 4 (radial vibrations).

Because the rig was lightly damped, data were acquired continuously during acceleration and deceleration of the rotor through its critical speeds. Data acquired consisted of the maximum amplitude at each frequency in the 0 to 500 Hz range as the rotor speed was accelerated from 4,000 to 7,200 r/min (66 to 120 Hz). The frequency ranges of interest for locating accessory vibrations and one, two and three per rev vibrations are shown in Figure 4.3. Output from the FFT analyzer was used by the control microcomputer to find peaks in the frequency ranges of interest, eliminate peaks below a threshold value based on the expected noise level for each frequency range, summarize the information in a tabular format required by the diagnostic system and transmit the data tables to the diagnostic system.



Range A:	1,000 - 3,500 rpm	Accessory Range
Range B:	4,000 - 7,200 rpm	1/rev Range
Range C:	12,850 - 13,400 rpm	2/rev Range
Range D:	19,600 - 20,000 rpm	3/rev Range

Figure 4-3

Frequency Ranges Used for Data Acquisition

4.3 ROTOR DIAGNOSTIC EXPERT SYSTEM

The diagnostic system was implemented as a nonconsultative system using the HARVEST fault tree analyzer [1] developed at MTI as the inference engine. HARVEST was selected because the knowledge base for the demonstration system was small enough to be represented as a fault tree, the inference engine and the interface to the data acquisition system could be modified to suit the problem and because the features available in HARVEST were adequate for the purposes of the demonstration. Creating a knowledge base required defining the fault tree using a tree definition language developed for HARVEST. The fault tree definition was then translated into a knowledge base for use by HARVEST using a translation program. Examples of the tree definition language are shown in Figure 4.4.

Data required for diagnosis was stored in data tables as shown in Figure 4.5. One table was used for each of the two channels in the FFT analyzer. The data consisted of frequency and amplitude peaks in the one per rev, two per rev, three per rev and accessory frequency ranges. Up to five peaks for each frequency range were stored.

Acceptable vibration amplitude limits for peaks in the frequency ranges of interest for data from channels A and B were stored in a limits table as shown in Figure 4.6. The limits were based on operational experience with the test rig. The critical speed range for the rotor, based both on an analysis of the rotor and actual performance of the rotor, were stored in a separate table as shown in Figure 4.6.

The symptoms for the five implanted faults are tabulated in Figure 4.7. The fault tree for the five implanted faults is shown in Figure 4.8. The part of the fault tree shown in Figure 4.8a is essentially the control and initialization section while the part shown in Figure 4.8b is the diagnostic logic using vibrations data obtained from the data acquisition system. The diagnostic logic is as follows: The data are first checked to determine if the increased support flexibility fault is present. If it is, the unbalance, misalignment and rub faults are not sought based on the assumption that the symptoms indicating those faults could be a result of a more flexible support. If the increased support flexibility fault is not present, the data are checked for unbalance followed by

```
LIMB OUTVAR

LIMB_LABEL:   BAL210

NEXT_LIMB:    QUE100

MESSAGE:

VARIABLES:    ROW:    COL:    UNITS:    NOTATION:
MESS04         7       5
MESS07         8       5
PK1_F1_A       8       50
MESS08         8       57
PK2_F1_A       9       50
MESS08         9       57

CLEAR_SCREEN:  YES

EXPLANATION:

COMMENTS:
PRINT UNBALANCE MESSAGE AND START RUB ANALYSIS


LIMB QUERY

LIMB_LABEL:  L1

QUESTION:
WHICH OF THE FOLLOWING WOULD YOU LIKE TO DO?

EXT_LIMB:     RESPONSE:
L2             QUIT
L3             DIAGNOSE WITH NEW VIBRATIONS DATA
L8             DIAGNOSE WITH EXISTING VIBRATIONS DATA

HORIZONTAL_MODE:  NO

CLEAR_SCREEN:  YES

EXPLANATION:

COMMENTS:
CONTROL LIMB TO SELECT FUNCTION
```

Figure 4-4

Examples of Limb Specification in the Tree Definition Language for HARVEST

Channel A

Interpolation None

Label	F0	A0	F1	A1	F2	A2	F3	A3
PK1	3150	0.963	6520	3.270	0	0.000	0	0
PK2	0	0.000	0	0.000	0	0.000	0	0
PK3	0	0.000	0	0.000	0	0.000	0	0
PK4	0	0.000	0	0.000	0	0.000	0	0
PK5	0	0.000	0	0.000	0	0.000	0	0
	1		2		3		4	

1. Frequency (F0) and Amplitude (A0) of peaks in accessory range
2. Frequency (F1) and Amplitude (A1) of peaks in 1/rev range
3. Frequency (F2) and Amplitude (A2) of peaks in 2/rev range
4. Frequency (F3) and Amplitude (A3) of peaks in 3/rev range

Figure 4-5

Vibration Data Table Format

VIBRATION LIMITS

LIMIT	ONEREV	TWOREV	THREEREV	ACC	RATIO
Probe A	10.0	0	0	0	0.12
Probe B	1.5	0.4	0.05	0.5	0

CRITICAL SPEED RANGES

MODE	WMIN	WMAX
Mode 1X	5200	5600
Mode 1Y	6400	6800

Figure 4-6

Vibration Limits and Acceptable Critical Speed Ranges for Demonstration Test Rig

Fault	Symptom	Data Required
Unbalance	Amplitude of any radial 1/rev peak greater than limit	Channel A (probe 2) Channel B (probe 4)
Misalignment	Ratio of the amplitude of axial 1/rev peak to radial 1/rev peak greater than limit	Channel A (probe 2) Channel B (probe 1)
Rub	Amplitude of radial 1/rev peak greater than limit, 2/rev peak greater than limit, and 3/rev peak greater than limit	Channel B (probe 4)
Increased Support Flexibility	Frequency of radial 1/rev peak less than the minimum critical speed for Mode 1X (Table 4)	Channel A (probe 2)
Accessory Vibrations	Amplitude of radial accessory peak greater than limit	Channel B (probe 4)

Figure 4-7

Symptoms of Implanted Faults

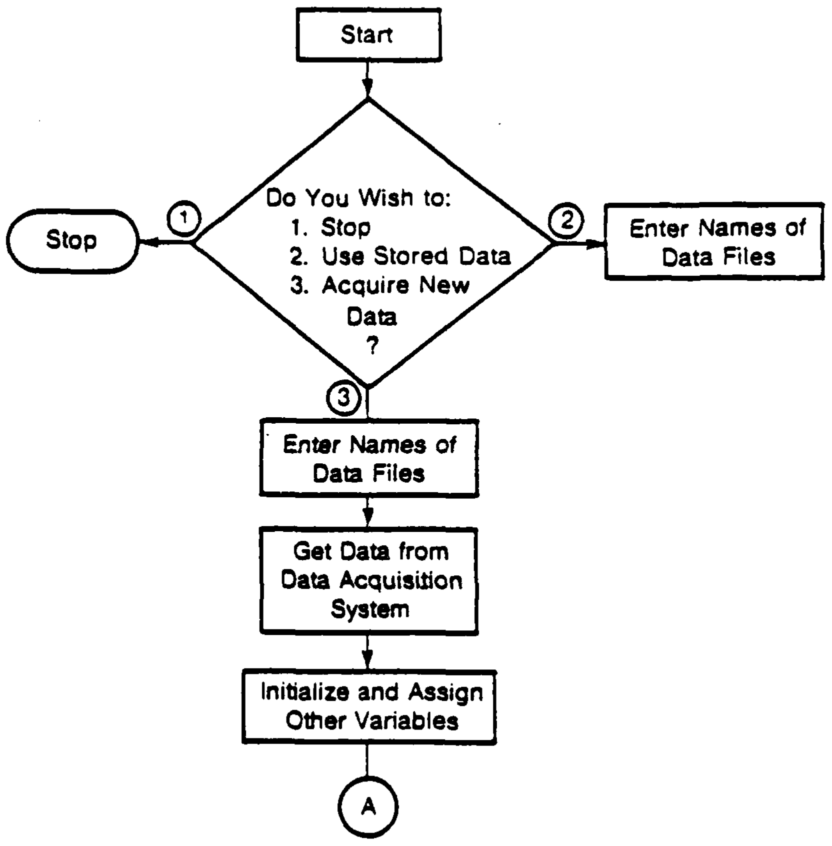

Start
Do You Wish to:
1. Stop
2. Use Stored Data
3. Acquire New Data
?
1
Stop
2
Enter Names of Data Files
3
Enter Names of Data Files
Get Data from Data Acquisition System
Initialize and Assign Other Variables
A

Figure 4-8a

General Flow Chart

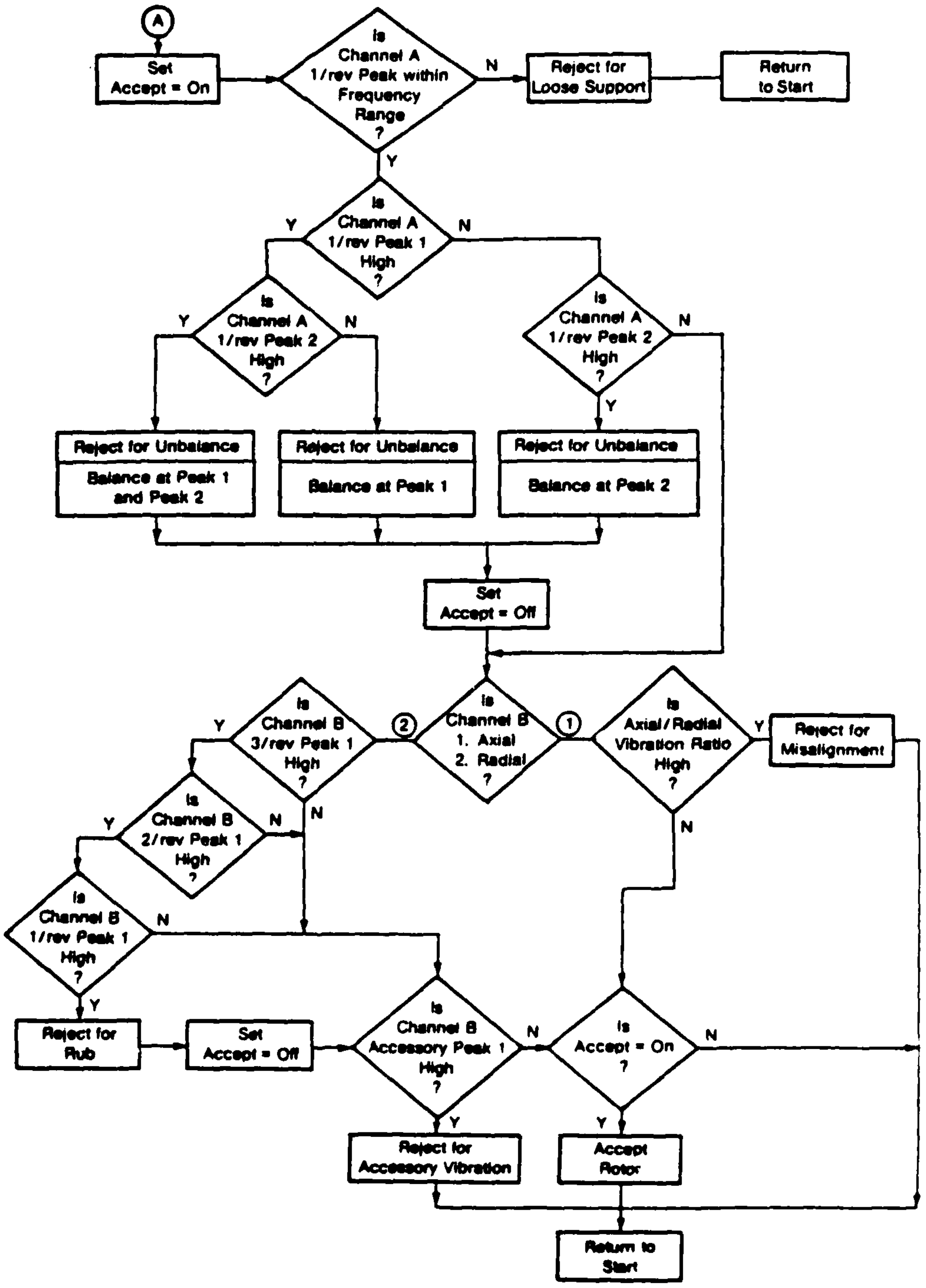

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Axial/Radial Vibration Ratio High ?
Y
N
Reject for Misalignment
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure 4-8b

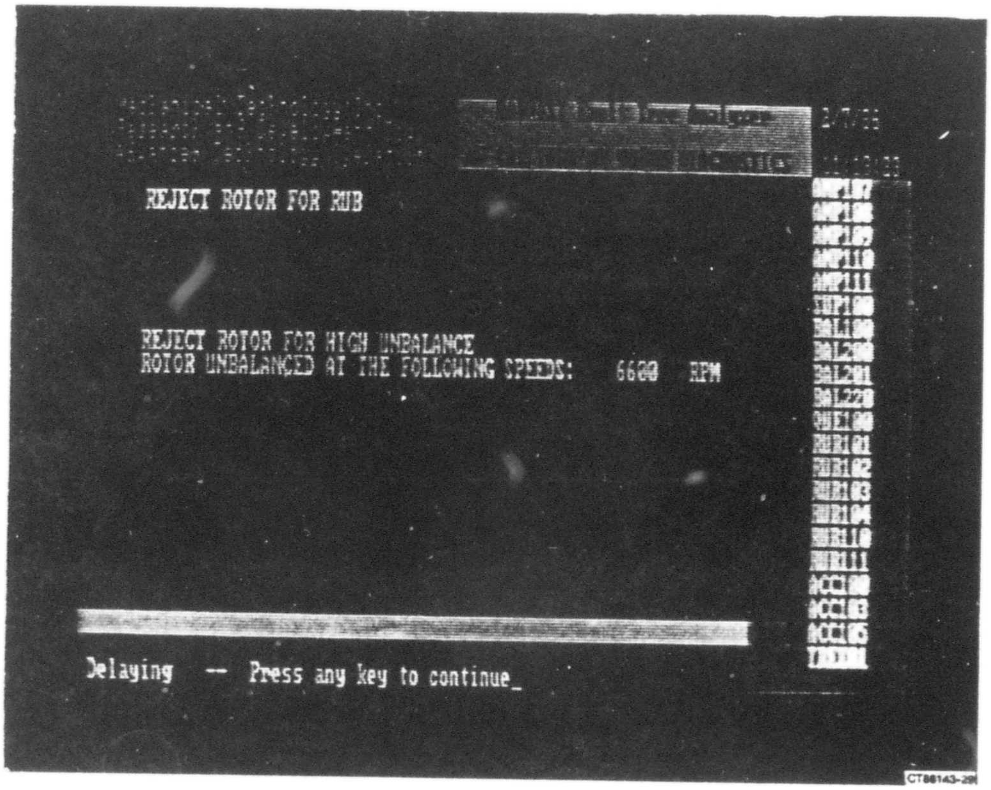
General Flow Chart

either rub and accessory faults(if channel B measures radial vibrations) or misalignment (if channel B measures axial vibrations). This restriction was required because the FFT analyzer only had two input channels and three probes were needed to diagnose all five faults in a single pass.

The fault tree as shown in Figure 4.8 was implemented and used successfully to detect the presence of any of the five faults implanted in the rotor. The data tables and results for each of the five faults are described in detail in Appendix B. A typical diagnostic session proceeds as follows: The diagnosis may be performed with data from a previous session stored in files or with new data. If the user elects to use old data, the expert system asks for the names of the data files and proceeds with the data analysis. If new data is to be used, the user is asked to input the names of the files in which the data will be stored for future use and the expert system sends a request for data to the data acqusition system. When the data acquisition system has acquired the required data, the data is sent to the expert system. The expert system then proceeds with the data analysis. If increased support flexibility fault is detected, the user is so informed and the diagnoses is stopped on the assumption that symptoms indicating other faults may have resulted from a loose or broken support. If the increase support flexibility fault is not detected, the user is asked whether channel B is configured for measuring axial or radial vibrations. The expert system then completes its analysis of data and informs the user of the results. The screen messages for a test case where both unbalance and rub were present are shown in Figure 4.9; the corresponding data tables are given in Figure 4.10 and the path through the fault tree is shown in Figure 4.11. The total time taken to acquire the data and perform the diagnosis was 10 minutes.

4.4 SUMMARY

An expert system based rotordynamic diagnostic system was successfully demonstrated by detecting faults implanted in a laboratory test rig. An assessment of gas turbine engine rotordynamic faults and associated symptoms showed that an expert system based diagnostic strategy can be developed for detecting and isolating faults in gas turbine engines. The placement of vibration sensors and assuring the integrity of vibrations data used to perform the diagnosis is crucial to the success of a diagnostic system. A sensor monitoring system to



```
REJECT ROTOR FOR RUB


REJECT ROTOR FOR HIGH UNBALANCE
ROTOR UNBALANCED AT THE FOLLOWING SPEEDS:   6600   RPM

Delaying   --  Press any key to continue_
```

```
AMP107
AMP108
AMP109
AMP110
AMP111
IUP100
BAL100
BAL200
BAL201
BAL220
QUE100
RUB101
RUB102
RUB103
RUB104
RUB110
RUB111
ACC100
ACC103
ACC105
[illegible]
```

Figure 4-9

Screen Display for HARVEST
Fault-Tree Analyzer

COMBINED FAULT - PROBE 2

LABEL	F0	A0	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6600.	17.800	13050.	3.080	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

COMBINED FAULT - PROBE 4

LABEL	F0	A0	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6600.	7.550	13200.	1.100	19650.	0.152
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

Figure 4-10

Data Tables for the Combined and Rub Fault

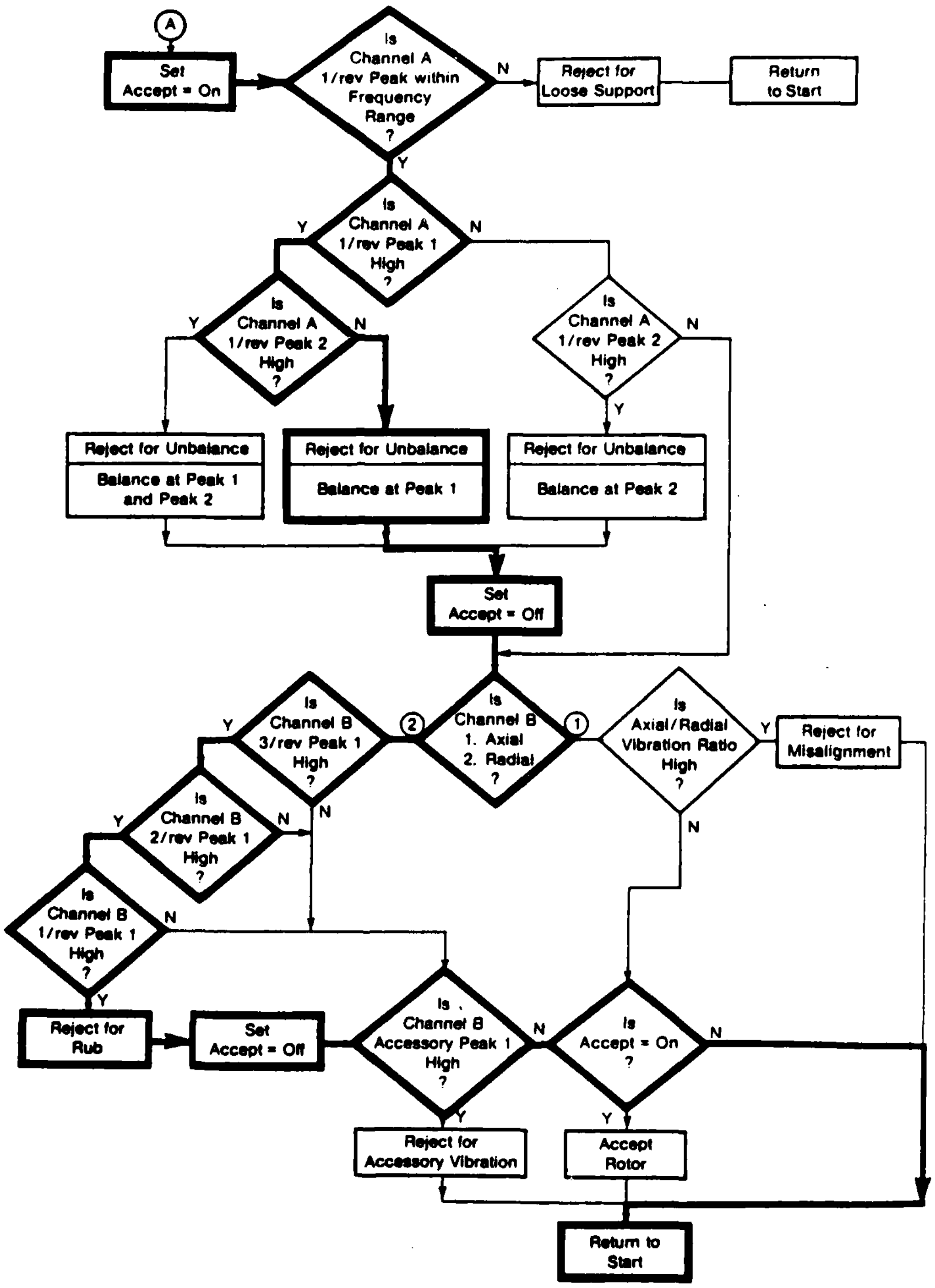

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Axial/Radial Vibration Ratio High ?
Y
N
Reject for Misalignment
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure 4-11

Path Through the Fault Tree for Combined Fault

detect and isolate sensor malfunctions should be an integral part of a rotordynamic diagnostic system.

4.5 REFERENCES

1. Karamchetty, Som D. S. R., " HARVEST - An Expert System", Proceedings of the ASME Computers in Engineering Conference (Volume Two), August 1985, pp. 243-248.

CHAPTER 5 PRELIMINARY SYSTEM SPECIFICATION AND IMPLEMENTATION PLAN (TASK 5)

The feasibility of using expert systems for rotordynamic fault detection was demonstrated in Task 4. The requirements for implementing an expert system based diagnostic system for gas turbine engine vibration problems are discussed below. A prototype system to prove the utility of an expert diagnostic system in actual use is recommended based on an assessment of the needs of UASF maintenance personnel.

5.1 ASSESSMENT OF USAF GAS TURBINE ENGINE MAINTENANCE NEEDS AND PRACTICES

The following assessment of USAF gas turbine engine maintenance needs is based on the information in references listed in Section 5.4. The information was analyzed to determine the key features required of an automated diagnostic system.

Maintenance strategy for USAF gas turbine engines is gradually being changed from practices based on operating time limits to an On-Condition Maintenance (OCM) strategy where the decision to remove an engine from an aircraft is based on a positive identification of deterioration in condition. For modular gas turbine engines such as the F100 engine, engine faults are identified to the smallest Line Replaceable Unit (LRU) and the malfunctioning LRU is then replaced. The success of OCM strategy is critically dependent on the ability of base level maintenance personnel to diagnose and isolate engine faults to LRU or module level. The maintenance personnel must then decide whether the engine can be used as is or if it should be removed from the aircraft to replace the faulty module. Decisions of this nature require experience and familiarity with the engines and an understanding of the environment in which they are used. Inadequate training and lack of experience will negate the advantages of OCM because engine modules may be replaced more often than necessary. In addition, the risk of engir: failures in flight increases because some modules may not be removed when they should be. Engine data must be interpreted correctly and consistently by maintenance personnel with different levels of skill.

OCM is also dependent on the availability of reliable engine maintenance history. For example, if LRUs or modules in a suspect engine are due for a scheduled maintenance of a life limited component, the engine may be removed from the aircraft even if the performance of the engine is acceptable. For modular engines, this implies keeping track of the performance and maintenance of individual modules as they are used with different engines. USAF experience has shown that maintenance history and engine performance trends can be used to predict imminent engine failures. A suspected fault in an engine should be evaluated with reference to the trends suggesting the fault. Currently engine maintenance data are acquired and processed manually, resulting in a considerable amount of paperwork and decreasing the timeliness and accuracy of data. An automated diagnostic system should be integrated into the maintenance data management system at base and depot level so that relevant data are available when needed. The system must also provide ways for tracking maintenance data when aircraft are deployed away from home base.

Current F100 engines rely on ground based diagnostic systems because most on-engine sensors were removed to minimize system weight and cost. Ground based diagnostics are accomplished with the engine installed in the aircraft. Maintenance actions are generally triggered by pilot "squawks". The field personnel then manually perform the procedures required to isolate problems using a diagnostic logic tree associated with each "squawk". If the problem is isolated, the LRU or module affected is replaced and the engine is returned to service after normal check runs. Approximately 45% of the squawks, however, cannot be duplicated on the ground and many are not repeated in flight when the engine is returned to service without any maintenance. The incidence of false alarms adds to the maintenance burden and must be reduced. Anomalous data caused by sensor malfunction or failure must also be detected and isolated; the sensors in a diagnostic system must not add to the maintenance burden.

An automated diagnostic system must take base level constraints such as manpower availability and level of skill into account. The aim should be to assist the users to perform the diagnosis by providing the relevant information. The maintenance personnel must be in the decision making loop. The number of steps required to arrive at a diagnosis should be minimized and the user must not be innundated with data. It is not enough for the diagnostic system to detect and isolate a fault: the system must recommend possible maintenance actions to return the engine back to flight status. If the recommended maintenance actions do not work as anticipated, the diagnostic system must reevaluate the situation based on any new information available. Maintenance history data should be updated to reflect the actual fault found and maintenance action taken.

The following features are considered important for an automated diagnostic system:

- Data compression to reduce processing time and ease the interpretation of data into suitable maintenance actions
- Sensors must provide reliable data in the engine environment anticipated during operational missions
- Repeatability of fault symptoms must be established in order to reduce false alarms
- The system must permit easy incorporation of new diagnostic logic and modification of existing logic
- The system must support bare base deployment
- Single point connections to engines installed in aircraft
- Diagnostic system malfunctions must be detected and isolated
- The system should be integrated into base and depot level maintenance history databases

5.2 APPLICATIONS FOR EXPERT DIAGNOSTIC SYSTEMS

The survey of USAF requirements and current practices suggest that expert diagnostic systems will be a cost effective solution because of the following reasons:

- Maintenance personnel training is not commensurate with the requirements for modern gas turbine engines
- Significant variations in levels of skill and experience of field maintenance personnel
- Loss of diagnostic expertise because of personnel turnover
- The need for the diagnostic system to be more than a data acquisition system. The system must identify faults and suggest possible maintenance actions

- Existence of diagnostic strategies for specific pilot "squawks"
- The need for a flexible and expansible system

The effectiveness of expert systems for diagnostics has been proven in applications such as medical diagnosis and automotive maintenance. The expertise of USAF maintenance personnel incorporated into an expert diagnostic system will provide a means to enable flight line maintenance personnel to perform their jobs at a level higher than they would without the expert system. The system will also provide the means to retain USAF diagnostic expertise when experienced maintenance personnel leave. Maintenance and diagnostic practices can be standardized using the best expertise available. The need for bulky manuals can be minimized by providing on-line explanation for the specific maintenance and diagnostic actions being performed; the system can therefore be used for on-the-job training. The architecture of expert systems (see Chapter 3) makes them inherently easy to modify and add to the knowledge contained in the system.

An integrated diagnostic system for gas turbine engines can best be implemented as a system with three layers: a monitoring and data acqusition system built into the engine, a portable maintenance system for flight line personnel and a diagnostic system for base level engine diagnosis and maintenance. The layered system can be phased in as individual elements are proven in prototype systems.

The monitoring and data acquisition system built into the engine should monitor the engine, use an expert system to detect faults, determine what data are needed to diagnose the fault and, finally, acquire and store the data for post flight analysis. By acquiring the data in flight under the conditions in which a fault occurs, the situations where pilot "squawks" cannot be duplicated will be reduced. The need to run the engine on the ground to acquire diagnostic data will also be reduced, conserving fuel and the life of other aircraft components which have to be powered up to run an engine on the ground. The issues of importance are the number, types and the placement of sensors, the volume of data to be stored and the scope of the diagnostic system. The impact on aircraft performance because of added weight and an increase in the initial cost of the engines have to be considered against reduced maintenance costs and improved availability of aircraft.

The portable maintenance system for flight line personnel should be a man portable, personal computer based system that has a single point connection for access to data acquired and stored in flight by the engine monitoring and data acquisition system. A communication facility to transfer data to and from the base level maintenance history database should be an integral part of the system. The system will use the in-flight data to perform a more thorough diagnosis, isolate the fault to the smallest LRU and recommend maintenance procedures. If needed, maintenance history data and data from additional tests may be used. On-line manuals for performing the recommended maintenance may be included to make the system self-sufficient for bare base deployment.

A more comprehensive gas turbine engine diagnostic system should be provided for base level engine shop. The diagnostic system should help the maintenance personnel decide if an engine/module/LRU that has been removed from an aircraft can be fixed at the base or has to be sent for depot maintenance. The scope of the diagnostic system will be determined by the repair capabilities at a base.

5.3 IMPLEMENTATION PLAN FOR PROTOTYPE SYSTEM

A prototype expert diagnostic system is discussed below to demonstrate the benefits of expert systems in operational use. The system should be designed for use at a base level engine repair facility for a modular engine such as the F100. The base level facility will provide a realistic proving ground while avoiding the risks inherent in incorporating a new technology into flight hardware.

The prototype system should diagnose suspect engines, engine modules or LRUs removed from aircraft on the flight line. Faults should be isolated to the smallest component/system that can be repaired or replaced at the base. Pertinent maintenance actions to repair and replace the component should be recommended. If repairs cannot be made at the base, the system should recommend shipping the engine/module/LRU for depot level maintenance. Other consideration for the system are:

1. Diagnostic Strategy: Diagnostic strategies need to be defined and verified. Current practices will serve as the starting point. USAF maintenance experts should be used to select the faults to be diagnosed

and develop diagnostic strategies for those faults. The strategies must be verified before they are implemented in a field system.

2. Sensors: The ability of sensors currently in use to provide the data required to implement the diagnostic strategies should be assessed. If additional sensors are required, the number, type and the placement of sensors should be determined. Design of the sensor complement should also address issues such as detection and isolation of sensor failures.

3. User Interface: Different user interfaces should be evaluated to determine the most effective means for the maintenance personnel to interact with the diagnostic system. Issues such as graphics vs text format, use of color vs monochrome displays and touch screen vs data entry should be considered.

Once the system design has been verified and the benefits assessed based on field testing, implementation plans for the full layered system can be developed. Application of the same diagnostic concepts to other rotating machinery such as APUs in USAF inventory should also be explored.

5.4 REFERENCES

1. Baker, L. E., De Hoff, R. L. and Hall, Jr., W. E., "Turbine Engine Fault Detection and Isolation Program - Phase I: Requirements Definition Study for an Integrated Engine Monitoring System", Technical Report AFWAL-TR-80-2053 (Vols 1 and 2) September 1980.

2. Weinstein, W. F., Corbin, T. W. and Weber, G. W., "Advanced Diagnostic Engine Monitoring System (ADEMS II)", Technical Report AFWAL-TR-80-2009 (Vols 1, 2 and 3) July 1980.

3. "USAF Updates Malfunction Detection Analysis System for C-5B Transport", Aviation Week and Space Technology, November 11, 1986, pp 203-205.

APPENDIX A

ROTORDYNAMIC TEST RIG

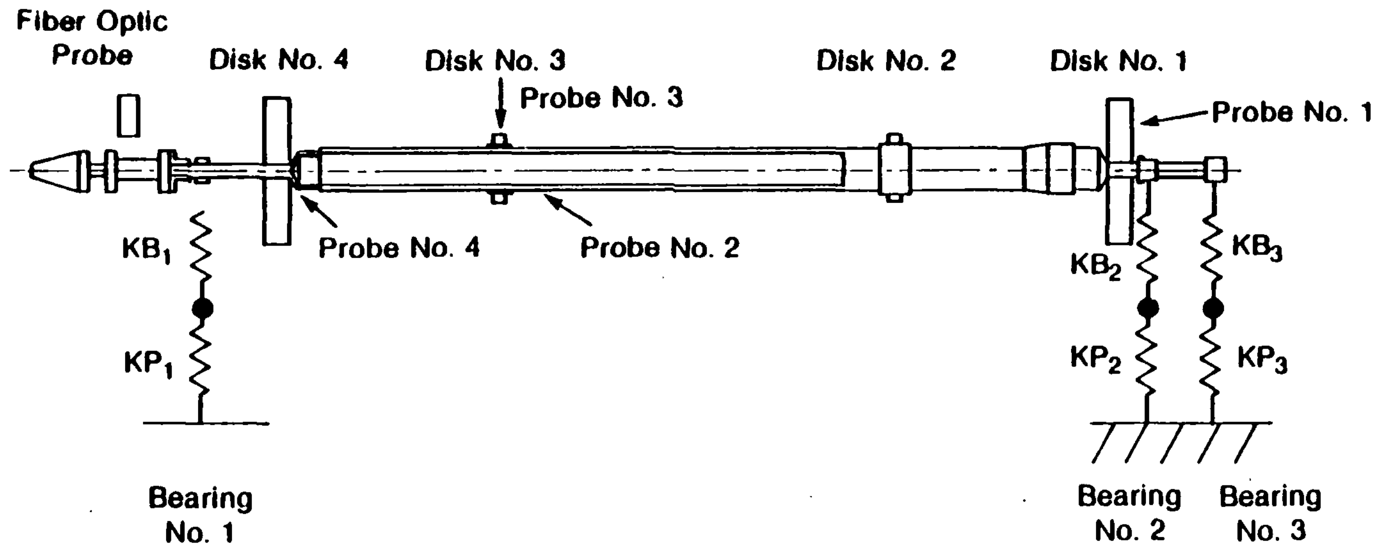
A.1 ROTOR HARDWARE

An existing MTI rotordynamic test rig was modified as required and used as a vehicle for conducting the demonstration tests. A schematic of the test rotor and bearing supports in the configuration used for the demonstration tests is shown in Figure A-1. The rotor of the selected rig consists of an aluminum tube with two steel stub shafts, one on each end of the aluminum tube. Steel disks were mounted on the stub shafts and the rotor was originally supported on two deep groove ball bearings mounted in rigid bearing pedestals. This rig was selected as the demonstration vehicle since it was capable of operating above one flexible critical speed, amenable to modification for implanting faults and exhibited repeatable operation.

To conduct the demonstration tests, it was necessary to implant known faults into the rotordynamic test rig. A list of potential faults was reviewed and several were chosen to be implanted in the rig. The following five faults were selected: Unbalance, Increased bearing support flexibility, Misalignment, Rub and Accessory vibration. The faults selected were deemed relevant to gas turbine rotor vibration diagnostics and it was felt that they could be successfully implanted in a rotordynamic test rig.

Introduction of the unbalanced rotor fault is predicated on the ability to first balance the rotor, and then disturb the state of balance. Previous experience with this test rig showed that it could be easily balanced using influence coefficient balancing methods, and that once balanced, the overall rig behavior was repeatable. Furthermore, the rig was known to have a response sensitivity at its critical speeds of approximately 10 mils/gram of weight in a balance plane. Thus, the unbalance fault was implanted in this rig by first balancing it at the appropriate speeds, and then adding weights of approximately 0.2 gram to one of the balance planes.

To implant the increased support flexibility fault in the rig, a bearing pedestal was designed which could be operated as rigid or flexible. A schematic of



Probe No. 1 — Axial Vibrations
Probe No. 2 — Horizontal Radial Vibrations
Probe No. 3 — Vertical Radial Vibrations
Probe No. 4 — Horizontal Radial Vibrations

Figure A-1

Schematic of a Test Rig

the bearing pedestal, which is used for the No. 2 bearing, is shown in Figure A-2. The pedestal shown in this figure can be operated as rigid or flexible by installing or removing the spacer respectively. For the flexure dimensions shown, the pedestal has a horizontal stiffness of approximately 10,000 lbs/in. This value of stiffness was chosen based on rotordynamics calculations.

A rotordynamic model of the rig was prepared and used to calculate the rotordynamic performance. The model used is shown in Figure A-3 and was first used to determine critical speeds for the baseline case - that is, no implanted faults. Next, the model was modified to reflect the effect of increasing the flexibility of the No. 2 bearing support. For this case, stiffness for this bearing support was 10,000 lbs/in. The results of critical speed calculations for both the baseline case and the increased flexibility case are shown in Table A-1. Both the forward and retrograde first critical speeds are shown. Generally, only forward critical speeds are of interest but, in this case the retrograde modes needed to be considered since the system is lightly damped and they may be responsive. Furthermore, for the increased flexibility case, which is a non-symmetric suspension, the mode in the direction of minimum stiffness (i.e. horizontal) is associated with the retrograde critical. As seen in Table A-1, the introduction of increased flexibility at the No. 2 bearing results in a significant change in the first flexible critical speed. For example, the retrograde critical in the increased flexibility case is below both the forward and retrograde critical of the baseline case.

The bearing pedestal shown in Figure A-2 was also designed to introduce static misalignment among the three rotor support bearings. In particular, this is accomplished by using an undersized spacer between the pedestal and the rigid support block. This spacer was made so that its thickness is less than the thickness of a spacer for which the three bearings are aligned (i.e., their centers fall on a straight line). Thus, when the undersized spacer is installed, the No. 2 bearing center is offset with respect to a straight line between the No. 1 and No. 3 bearing centers.

A static misalignment analysis of the shaft was performed to assess the maximum shaft deflection for misalignment of the No. 2 bearing. This analysis treats the shaft as a (redundant) beam and determines shaft deflection and bearing loads for prescribed displacements. The rotor model shown in Figure A-3 was

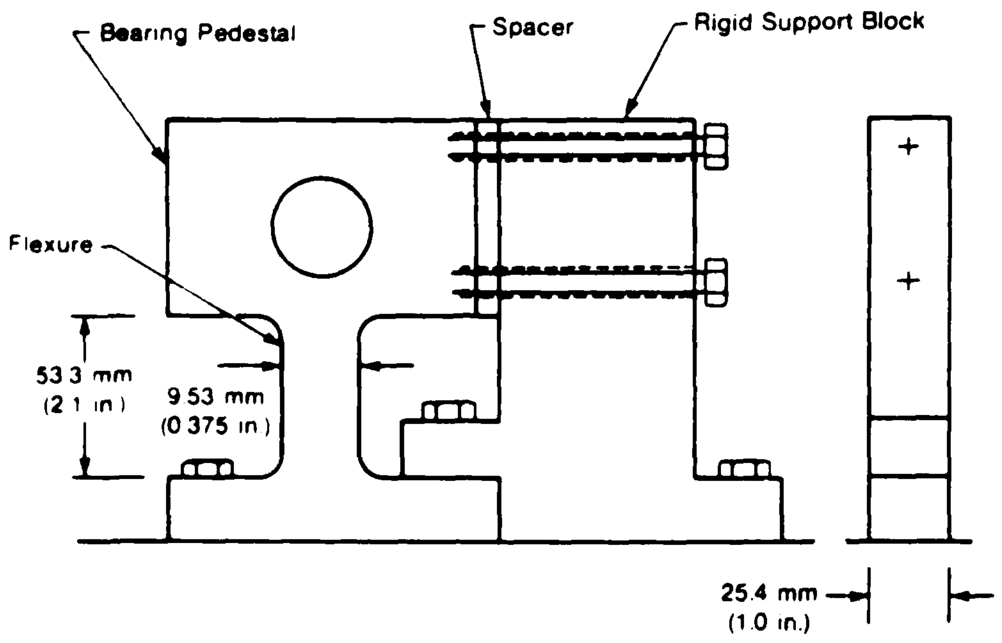


Figure A-2

Number 2 Bearing Pedestal

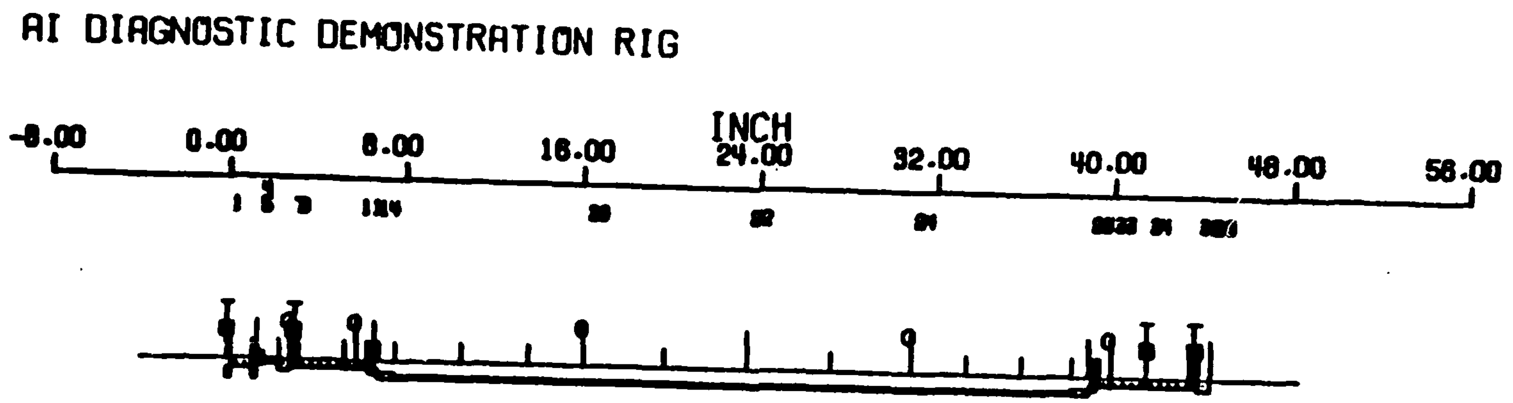


Figure A-3

Rotordynamic Model

Table A-1

PREDICTED FIRST CRITICAL SPEEDS (R/MIN)

	RETROGRADE	FORWARD
BASELINE CASE (3 Rigid Bearings)	5,463	6,546
INCREASED FLEXIBILITY (No. 2 Bearing in flexible support)	4,380	5,098

used, and a plot showing static shaft deflection versus axial length is shown in Figure A-4. As seen in this Figure, the maximum shaft deflection is approximately 2.5 times the radial misalignment of the No. 2 bearing (which in this case was prescribed to be 10 mils). Thus, this method of introducing misalignment provides significant static shaft bending. Furthermore, for 10 mils of No. 2 bearing misalignment, loads at all three bearings were below their load limit. Therefore, this approach was selected for introducing misalignment and specific radial deflection to be imposed was determined at the time of test.

To introduce the rubbing fault, it is necessary to impose controlled contact between the rotor (while rotating) and a stationary component. Figure A-5 is a schematic of the cross-section of the No. 4 rotor disk and stationary bracket. As shown, the bracket has a threaded hole which accepts a bolt secured by a jam nut. The bolt is initially installed with a known clearance between it and the disk "high spot". A rub fault occurs when the rotating disk comes in contact with the bolt. This contact occurs when the disk orbit amplitude exceeds the bolt/disk clearance. To control the speed at which the rub occurs, the bolt was installed with approximately 1.5 mils of clearance, and the rotor was unbalanced so that disk orbit exceed the clearance at a speed just below the first critical.

Rather than running the rig with a faulty accessory, the accessory vibration fault was simulated with an external excitation system. Specifically, an electrodynamic shaker simulates the effect of an accessory vibration that is, vibration of stationary components at a noninteger ratio of the rotor critical speed. To accomplish this a stinger is connected between the shaker and the stationary bracket as shown in Figure A-5. The equipment used with the shaker system is shown in Figure A-6. Here, the frequency of the external vibration is controlled by the oscillator. For these tests, the oscillator frequency was set at approximately 0.47 times the critical speed.

A.2 INSTRUMENTATION

To measure rig dynamic behavior, several vibration measuring sensors were required. For this rig, instrumentation consisted of four proximity probes and proximiters which measure vibration. A fiber optic sensor was used for measuring speed and phase reference.

MISALIGNMENT

3 BEAR & PEDESTALS - (INCREASED FLEX AT PEDESTAL # 2)

SYSTEM COND. 1
SPEED CASE 1
LEVEL NUMBER 1

X-AMP ○
Y-AMP △

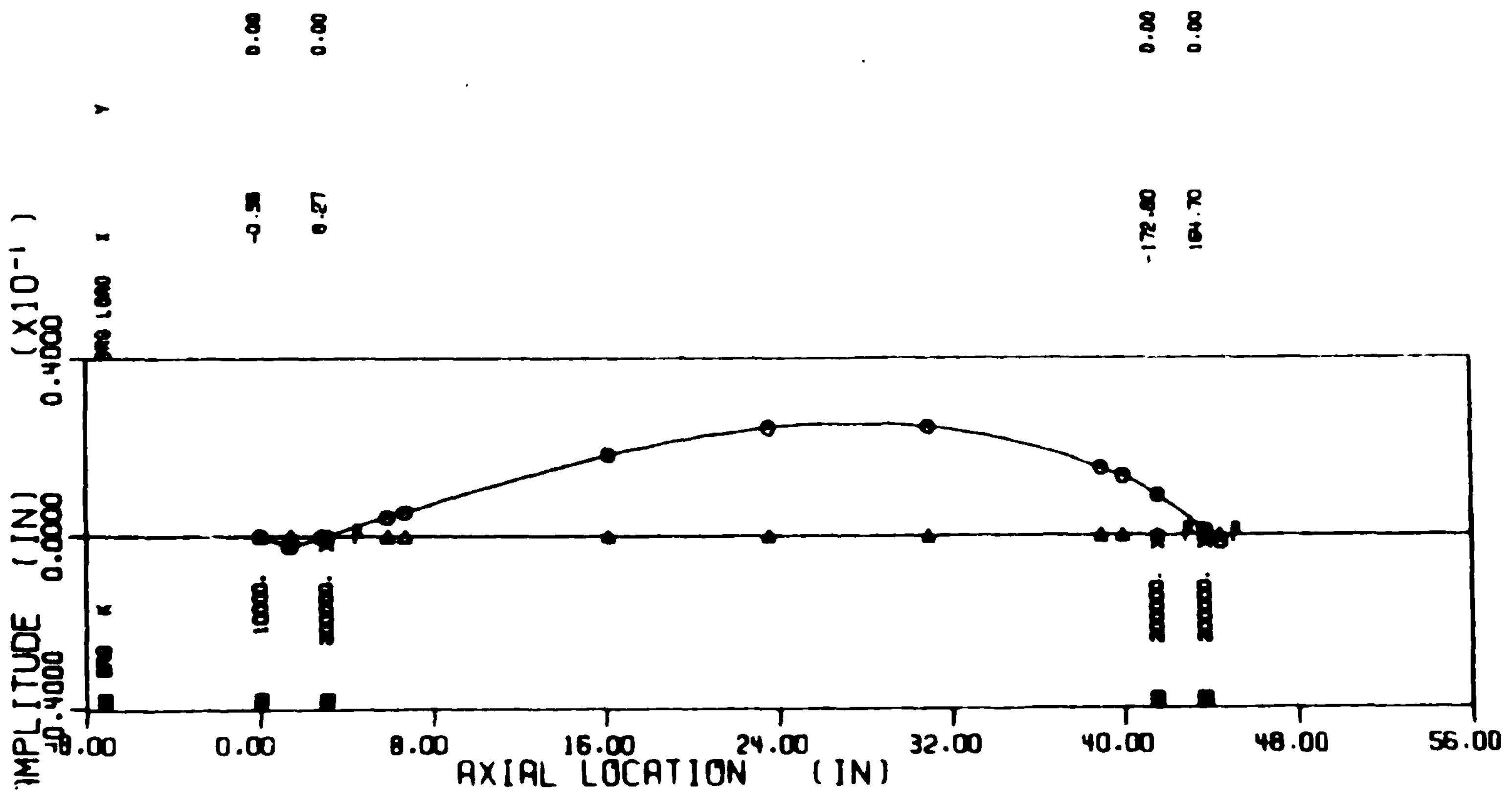


Figure A-4

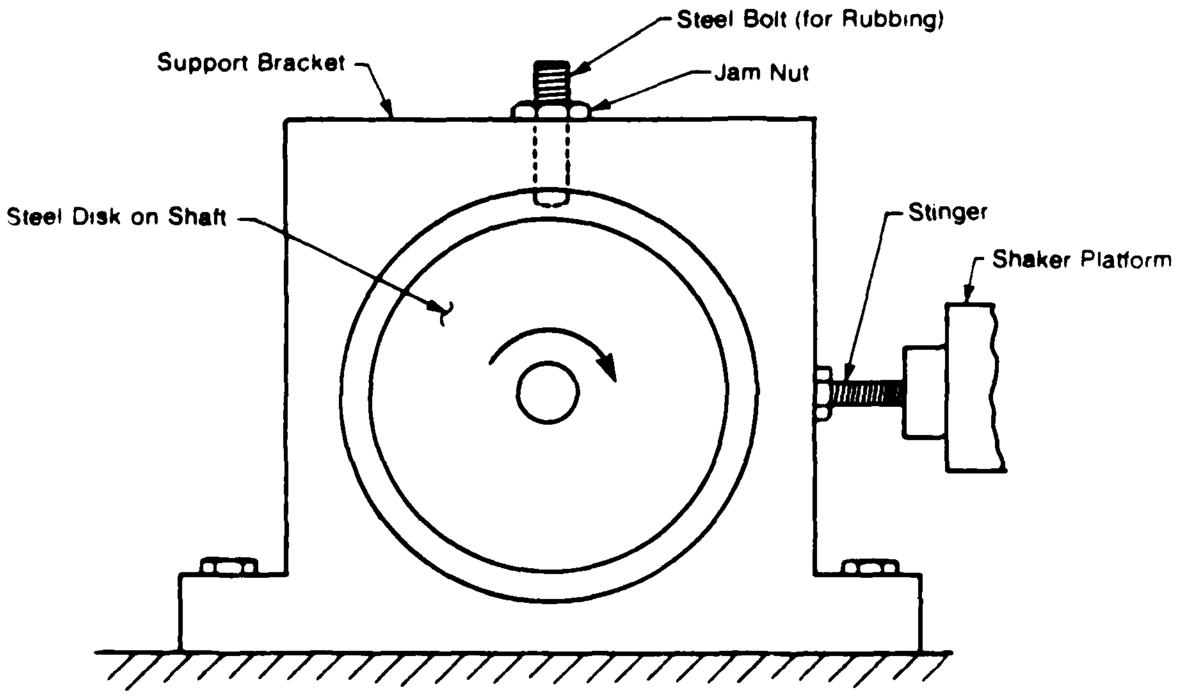


Figure A-5

Number 4 Disk and Bracket Cross-Section

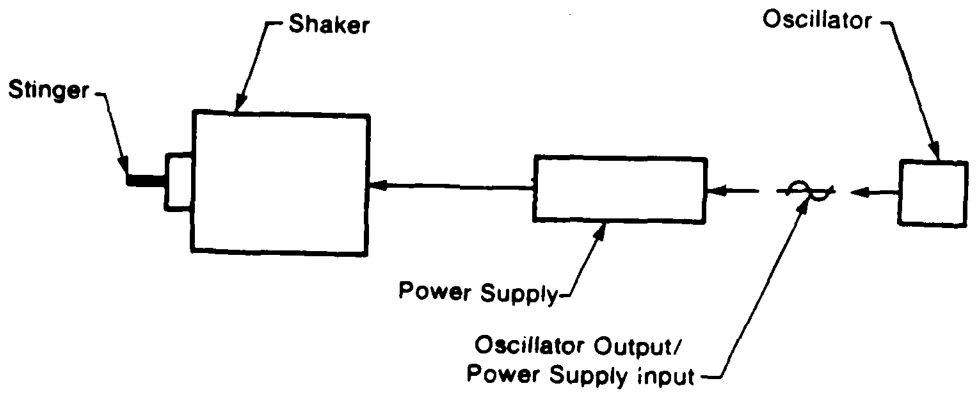


Figure A-6

Shaker System Hardware

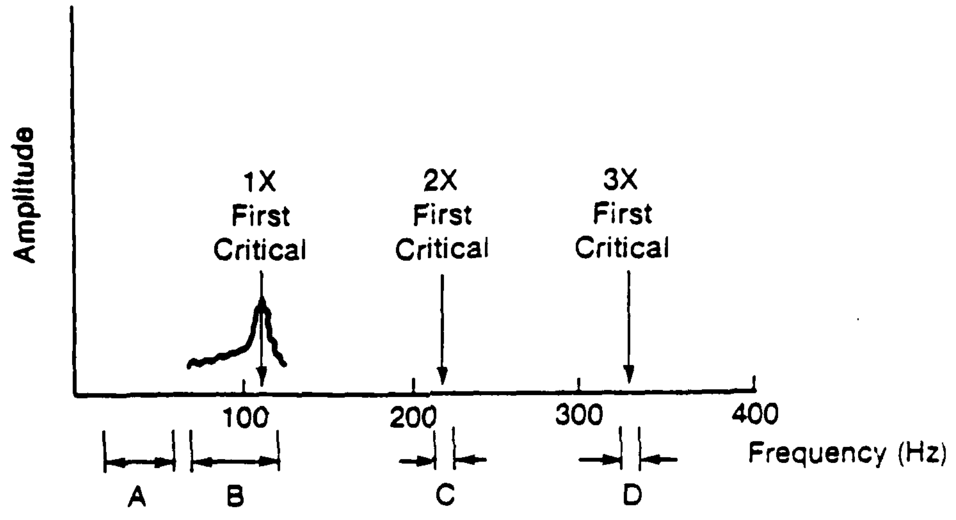
Vibration measuring sensors were located to measure response caused by the implanted faults. Figure A-1 shows the location of the four proximity probes. At the first critical speed, the shaft response is maximum at its center in the baseline case. Furthermore, for the increased flexibility case, response is maximum in the horizontal direction at the shaft center. Probe number 2 is used for both the baseline and increased flexibility cases since it is the probe closest to the center and it measures response in the horizontal direction. For rub and external excitation, response is maximum at the number 4 disk, therefore, probe 4 is used for identification of these faults. It was necessary to measure shaft axial response to identify the misalignment fault. Probe number 1 was used for this purpose.

A.3 DATA ACQUISITION

Data acquisition was controlled by an HP-9816 microcomputer. When a request for data is received from the expert system, the controller directs the collection and processing of vibrations data from the probes. Signals from the appropriate probes on the rotor system were fed to an Ono Sokki FFT Analyzer (Model CR-920) to convert sensor signals into frequency domain data comprising of frequency and amplitude information. Due to the fact that the demonstration test rig was lightly damped, it was necessary to continuously acquire rig response data during its acceleration (or deceleration) through critical speeds. Data was acquired while accelerating the rig and operating the analyzer in the following manner:

- Spectrum mode (i.e., amplitude vs frequency)
- Peak store mode (i.e., stores maximum amplitude at each frequency)
  - Start store mode at rig speed of ≃ 4,000 r/min
  - Stop store mode at rig speed of ≃ 7,200 r/min

Spectrum data acquired in this manner were used to identify one-, two- and three-per-rev vibration components, and vibration at other frequencies. The data in the spectrum were separated into different ranges to obtain the pertinent amplitudes as shown in Figure A-7. For example, the data between 66.7 hz and 120 hz (4,000 to 7,200 r/min) contain the one-per-rev amplitudes since the rig was operated in this range while the analyzer was in store mode. If they



Range A:	1,000 - 3,500 rpm	Accessory Range
Range B:	4,000 - 7,200 rpm	1/rev Range
Range C:	12,850 - 13,400 rpm	2/rev Range
Range D:	19,600 - 20,000 rpm	3/rev Range

Figure A-7 Frequency Ranges Used for Data Acquisition

exist, the two- and three-per-rev peaks of the critical occur near 220 and 330 hz respectively since the first critical is at approximately 110 hz (6,600 r/min). The actual frequency ranges used for finding two- and three-per-rev peak amplitudes are shown in Figure A-7. Finally, the amplitude at other frequencies was needed and for this rig was limited to frequencies in the 15-60 hz range.

The output from the FFT analyzer was used by the controller to find peaks in the appropriate frequency ranges, eliminate peaks below a threshold value, summarize the information in tables suitable for the expert system and transmit the data tables to the expert system.

APPENDIX B

DIAGNOSIS OF IMPLANTED FAULTS IN THE DEMOSTRATION TEST RIG

B.1 INTRODUCTION

Five faults were implanted in the demostration test rig: unbalance, misalignment, increased support flexibility, rub and accessory vibration. The data for each of the five faults is described below. The data include frequency and amplitude plots for the baseline case with no implanted fault and with an implanted fault, the data tables sent to the expert system and the fault tree showing the diagnostic path followed by the expert system. Table VIB-A contains channel A vibrations data from probe 2 (see Appendix A) while Table VIB-B contains channel B vibrations data from either probe 1 (axial vibration) or probe 4 (radial vibration). Both tables use the same format shown in Figure B-1. Other tables used in the diagnosis are the vibration limits table (VIBLIM) and the critical speed range table (CRIT). Table VIBLIM has the vibration limits for each of the four frequency ranges of interest. Table CRIT has the acceptable critical speed ranges for one-per-rev vibrations. The two tables for the demonstration test rig are shown in Table B-1.

B.2 BASELINE CASE WITH NO IMPLANTED FAULTS

The rig was first operated in the baseline condition (i.e., with no implanted faults). To get the rig in this condition, it was assembled as follows:

- Bearing No. 2 in rigid condition
- Bearing No. 2 aligned
- Rub bolt not installed
- External excitation not activated

With the rig assembled in this manner, it was balanced using the influence coefficient balancing method and a series of baseline runs were made using the instrumentation and data acquisition hardware described in Appendix A. Figure B-2 shows speed sweep data acquired during rig acceleration for probes 2 and 4 with the rig in the baseline condition. Note that for each probe there is a peak at 6,520 r/min with amplitudes of 6.48 and 2.75 mils for Probe 2 and 4 respec-

Channel A

Interpolation None

Label	F0	A0	F1	A1	F2	A2	F3	A3
PK1	3150	0.963	6520	3.270	0	0.000	0	0
PK2	0	0.000	0	0.000	0	0.000	0	0
PK3	0	0.000	0	0.000	0	0.000	0	0
PK4	0	0.000	0	0.000	0	0.000	0	0
PK5	0	0.000	0	0.000	0	0.000	0	0
	1		2		3		4	

1. Frequency (F0) and Amplitude (A0) of peaks in accessory range
2. Frequency (F1) and Amplitude (A1) of peaks in 1/rev range
3. Frequency (F2) and Amplitude (A2) of peaks in 2/rev range
4. Frequency (F3) and Amplitude (A3) of peaks in 3/rev range

Figure B-1

Vibration Data Table Format

Table B-1

VIBRATION LIMITS AND ACCEPTABLE CRITICAL SPEED RANGES FOR DEMONSTRATION TEST RIG

VIBRATION LIMITS

LIMIT	ONEREV	TWOREV	THREEREV	ACC	RATIO
Probe A	10.0	0	0	0	0.12
Probe B	1.5	0.4	0.05	0.5	0

CRITICAL SPEED RANGES

MODE	WMIN	WMAX
Mode 1X	5200	5600
Mode 1Y	6400	6800

BASELINE
(PROBE 2)

BASELINE
(PROBE 4)

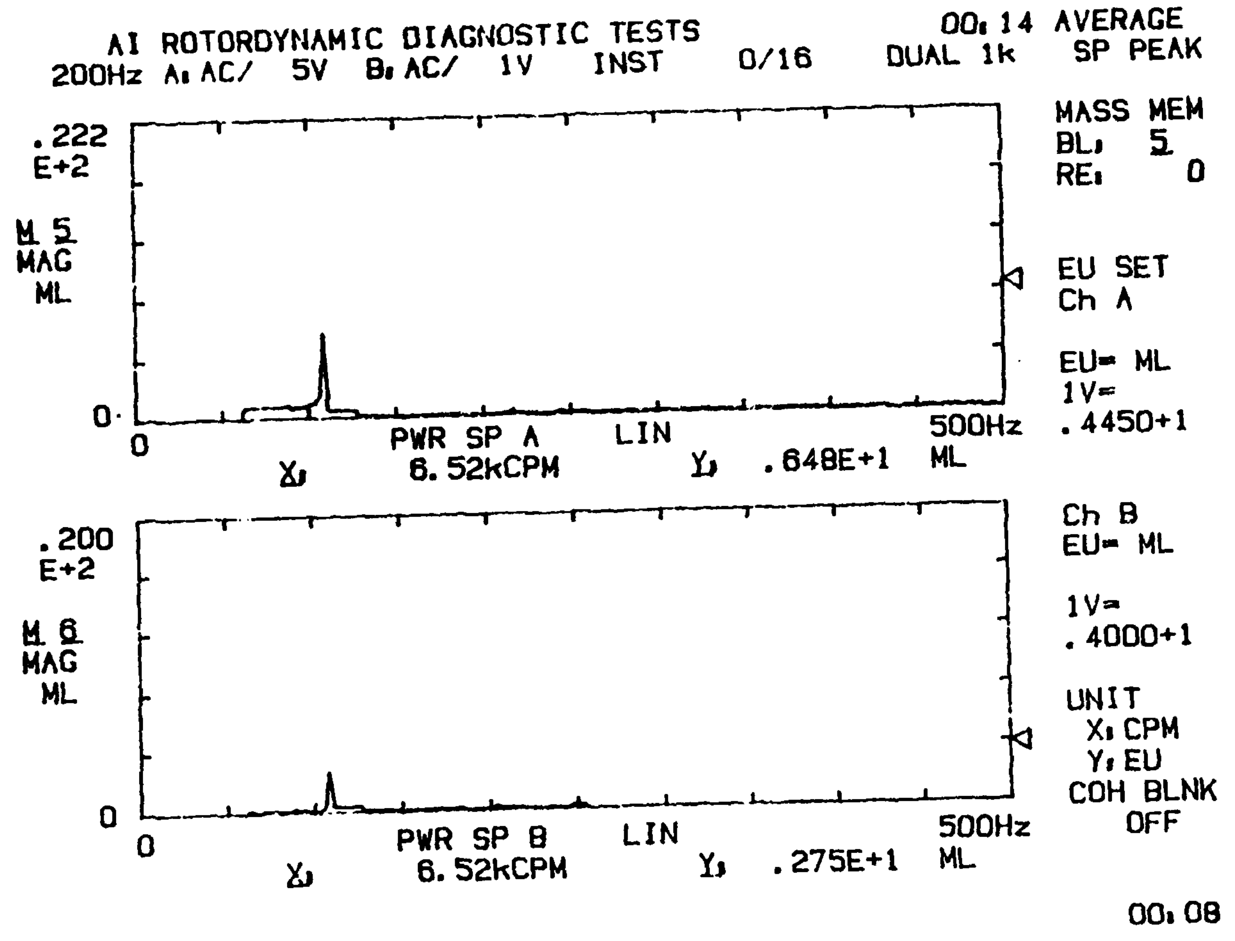


Figure B-2

tively (amplitude and frequency of the highest peak in each plot are shown directly below the abcissa labels as X and Y respectively). In each case, the peak is the first critical speed, and agrees well with predictions. Furthermore, the peak at 6,520 r/min is the only significant component for both Probe 2 and 4. Data tables for the baseline case are shown in Table B-2 and the path through the fault tree is shown in Figure B-3. Note that only the part of the fault tree that determines the fault based on vibrations data is shown in Figure B-3; the control part of the tree has not been shown for the sake of brevity. The path through the fault tree can be traced by comparing the vibration amplitudes in the data tables with the in Table B-1.

B.3 UNBALANCE

As seen in Figure B-2, the maximum probe 2 one-per-rev response in the balanced state is approximately 4.6 mils and occurs at the first critical (6,600 r/min). Unbalance was introduced by adding 0.2 grams to disk no. 2 and Probe 2 was used to identify the unbalance fault. When the rotor was unbalanced, its response was significantly higher than for the baseline case. Therefore, the vibration limit was set to 10 mils in the one-per-rev range for probe 2.

Figure B-4 shows probe 2 response for both the baseline case and the unbalance fault. With the rig unbalanced, the maximum rig speed was 6,520 r/min and it was not possible to traverse the first critical or complete the speed sweep to 7,200 r/min. In particular, as seen in Figure B-4, the one-per-rev amplitude at maximum speed was 11.4 mils. The ES diagnostic logic recognizes the one-per-rev amplitude greater than the 10 mil limit at (or near) the first critical to identify the unbalance rotor condition. Table VIB-A for the unbalance fault is shown in Table B-3 and the path through the fault tree in Figure B-5.

B.4 MISALIGNMENT

Generally speaking, misalignment manifests itself in two ways with respect to rotor vibration. They are:

- Increase in two-per-rev (radial) response
- Increase in the ratio of one-per-rev axial to radial response

Table B-2

TABLES V1B-A and VIB-B FOR BASELINE CASES

VIB-A

BASELINE - PROBE 2
INTERPOLATION NONE

LABEL	FO	AO	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6520.	6.480	13050.	.470	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

VIB-B

BASELINE - PROBE 4
INTERPOLATION NONE

LABEL	FO	AO	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6520.	2.750	0.	0.000	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

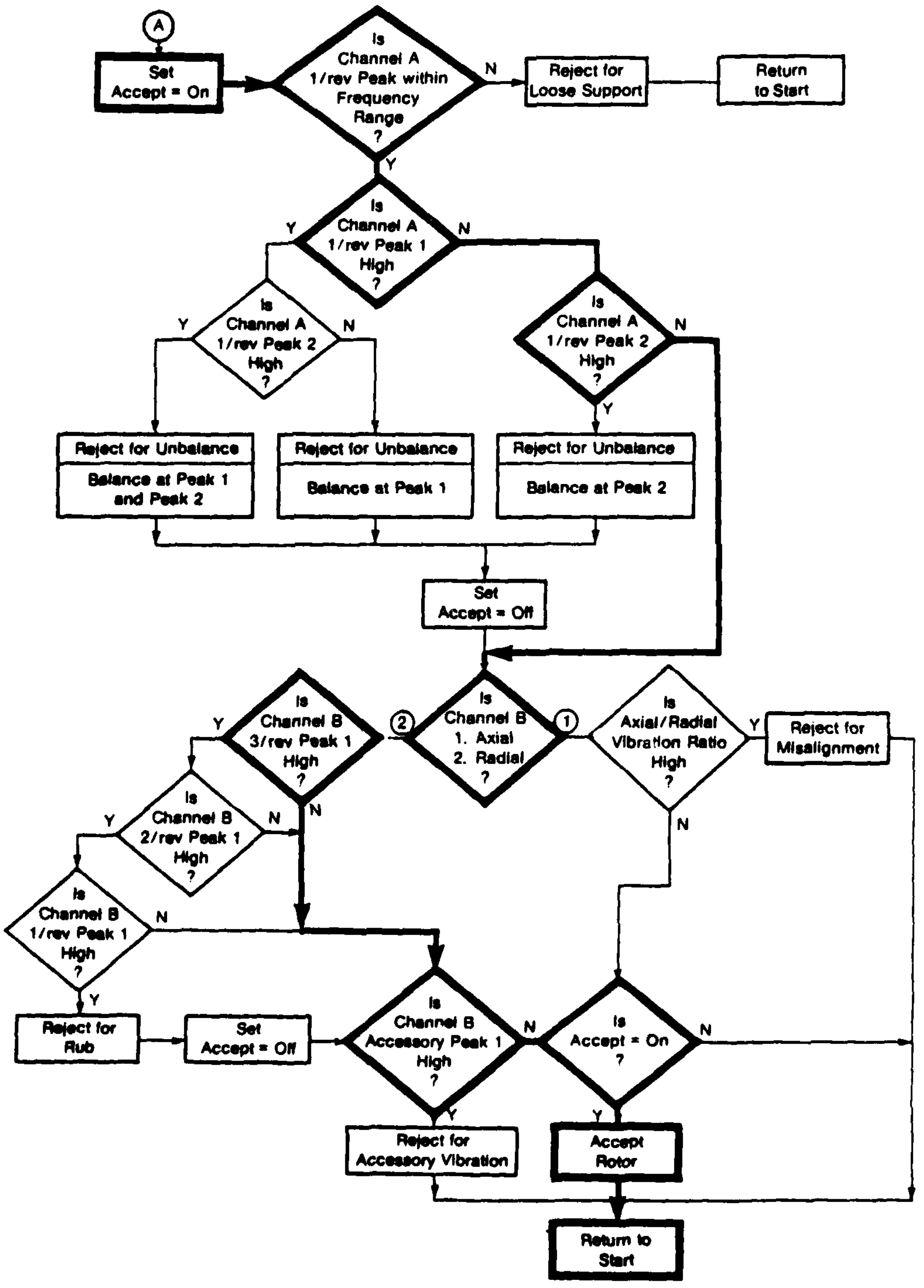

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Y
Reject for Loose Support
Return to Start
Is Channel A 1/rev Peak 1 High ?
Is Channel A 1/rev Peak 2 High ?
Is Channel A 1/rev Peak 2 High ?
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
1
2
Is Axial/Radial Vibration Ratio High ?
Reject for Misalignment
Is Channel B 3/rev Peak 1 High ?
Is Channel B 2/rev Peak 1 High ?
Is Channel B 1/rev Peak 1 High ?
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
Is Accept = On ?
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-3 Path Through the Fault Tree for No Faults

BASELINE
(PROBE 2)

UNBALANCE
(PROBE 2)

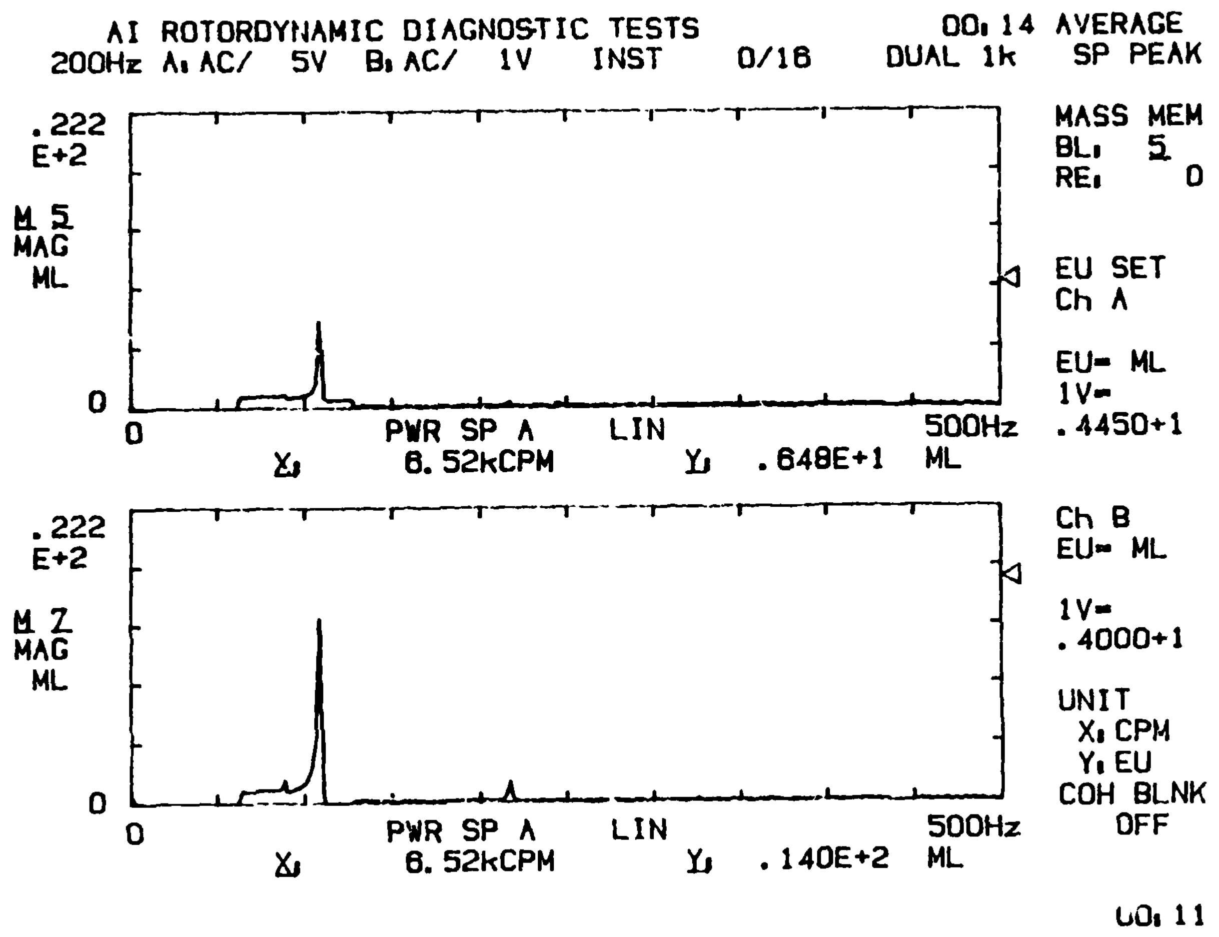


Figure B-4

Table B-3

TABLE VIB-A FOR UNBALANCE

```
UNBALANCE - PROBE 2
INTERPOLATION NONE
LABEL   FO      AO       F1      A1       F2      A2      F3       A3

PK1     0.    0.000   6520.   14.000  13050.   1.580    0.       0.
PK2     0.    0.000      0.    0.000      0.   0.000    0.       0.
PK3     0.    0.000      0.    0.000      0.   0.000    0.       0.
PK4     0.    0.000      0.    0.000      0.   0.000    0.       0.
PK5     0.    0.000      0.    0.000      0.   0.000    0.       0.
```

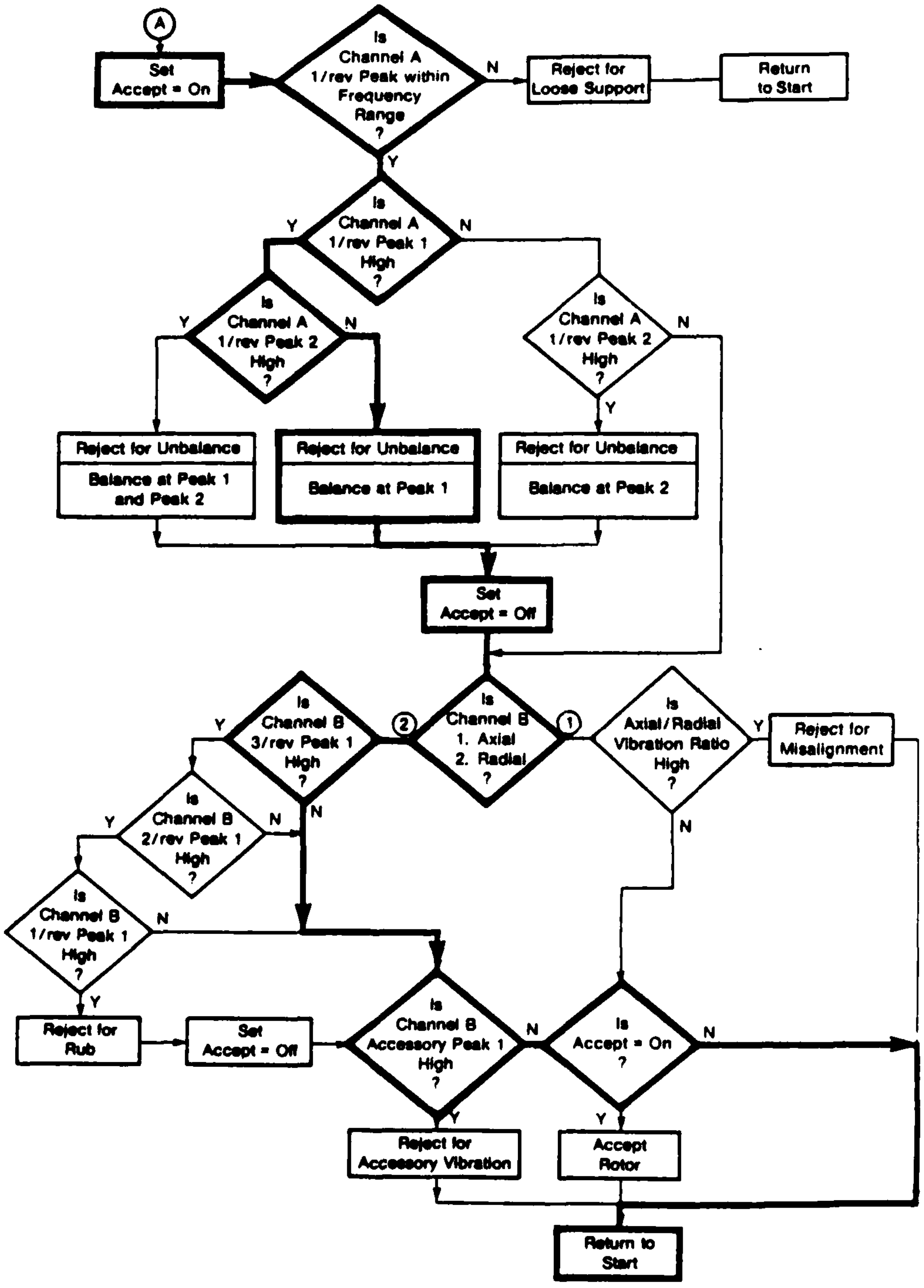

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 3/rev Peak 1 High ?
Y
2
Is Channel B 1. Axial 2. Radial ?
1
Is Axial/Radial Vibration Ratio High ?
Y
Reject for Misalignment
N
Is Channel B 2/rev Peak 1 High ?
Y
N
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Is Accept = On ?
N
Y
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-5

Path Through the Fault Tree for the Unbalance Fault

The increase in two-per-rev response due to misalignment is most evident when the rotor is operating at or near its rigid body critical speeds. As noted earlier, the test rig used for this demonstration operates above its first flexible critical speed, and in fact, has no rigid body critical speeds with the bearing suspension used. Since this rotor system operates above its first flexible critical, the misalignment fault was identified through the change in the ratio of axial to radial response.

Misalignment was introduced by installing a spacer block in the No. 2 bearing pedestal that was 8 mils thinner than the the nominal spacer block resulting in a radial shift of the bearing center with respect to the straight line formed by the other two bearing centers. Figure B-6 shows speed sweep data for Probe No. 1 which measures axial response. The baseline case (upper plot in Figure B-6) shows a small one-per-rev amplitude at the first critical which, in fact, is only slightly above run out. On the other hand, with misalignment imposed (lower plot), the one-per-rev is approximately 50% greater than the baseline case. This increase in one-per-rev axial response results in an increase in the axial to radial response ratio which the expert system uses to identify misalignment. Tables VIB-A (radial response) and VIB-B (axial response) for misalignment are shown in Table B-4; the axial to radial response ratio of 0.16 and is greater than the acceptable limit of 0.12. The path through the fault tree is shown in Figure B-7.

B.5 INCREASED SUPPORT FLEXIBILITY

Probe 2 was used to identify increased flexibility fault which was introduced by operating the bearing No. 2 pedestal in its flexible configuration. Figure B-8 shows speed sweep data for probe 2 for both the baseline case and with increased flexibility. Note that with increased flexibility (lower trace) the maximum attainable speed was 4,350 r/min and the one-per-rev amplitude at this speed is approximately 11 mils. This is due to the fact that the critical speed dropped (due to increased flexibility) causing large amplitudes at a speed well below the baseline case first critical. It is this change in the location of the one-per-rev peak that the ES identifies to diagnose increased flexibility. Table VIB-A for increased support flexibility is shown in Table B-5 and the path through the fault tree is shown in Figure B-9.

BASELINE
(PROBE 1)

MISALIGNMENT
(PROBE 1)

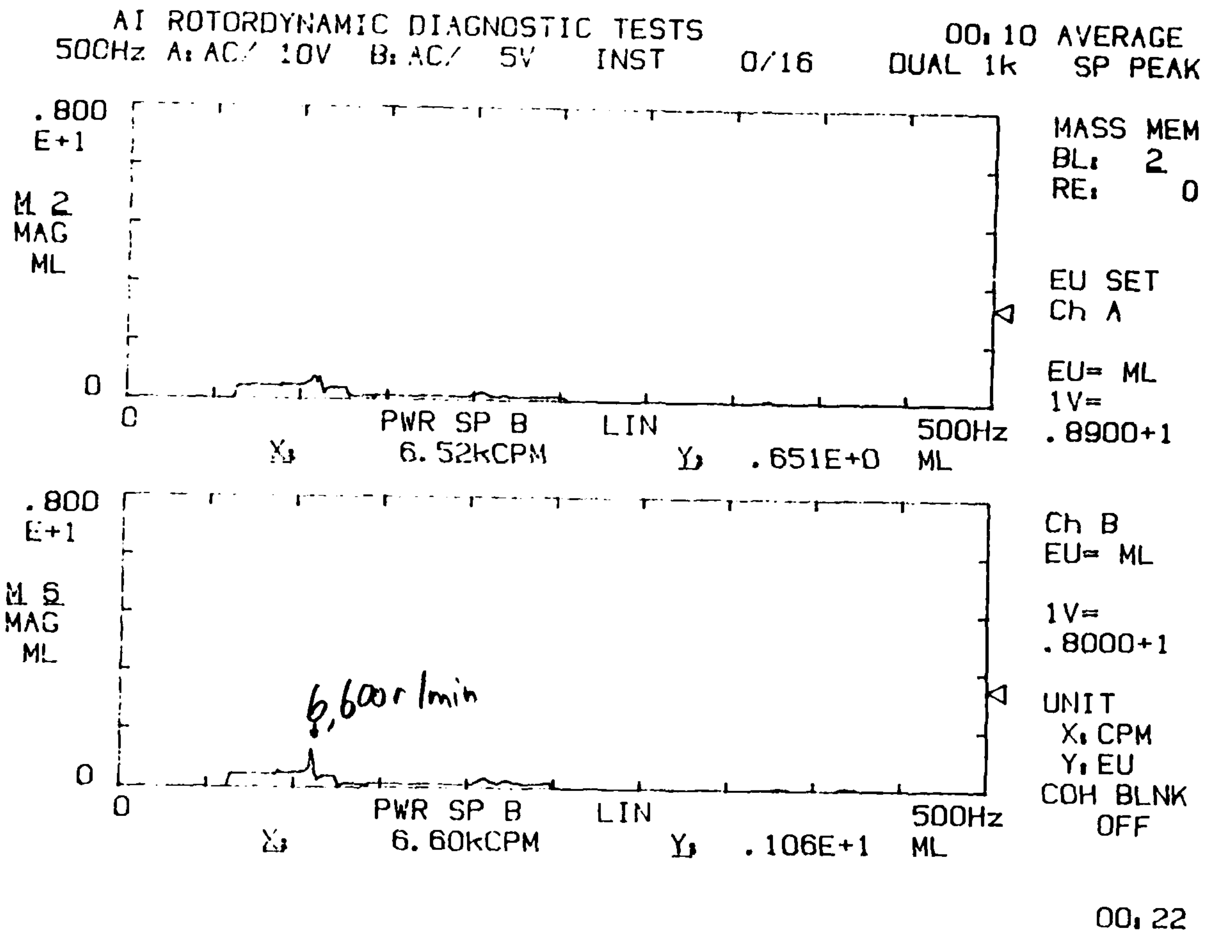


Figure B-6 - Misalignment

Speed Sweep (4,000-7,000 r/min) Baseline

Table B-4

TABLES VIB-A AND VIB-B FOR MISALIGNMENT

MISALIGNMENT - PROBE 2
INTERPOLATION NONE

LABEL	FO	AO	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6600.	6.640	13270.	.463	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

MISALIGNMENT - PROBE 1
INTFRPOLATION NONE

LABEL	FO	AO	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6600.	1.050	0.	0.000	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

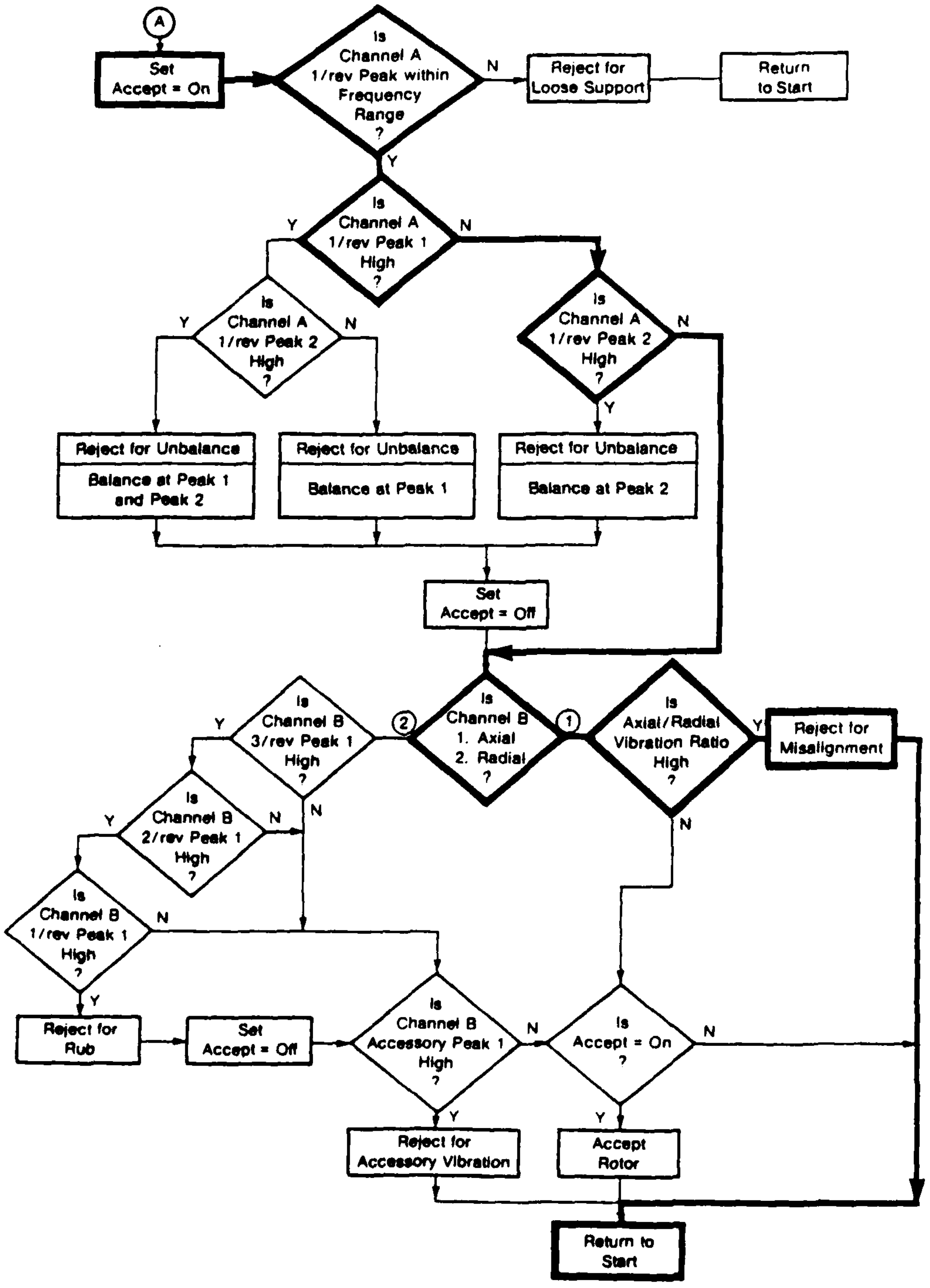

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Y
Reject for Loose Support
Return to Start
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Axial/Radial Vibration Ratio High ?
Y
N
Reject for Misalignment
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-7 Path Through the Fault Tree for the Misalignment Fault

BASELINE

(Probe 1)

INCREASED FLEXIBILITY

(Probe 2)

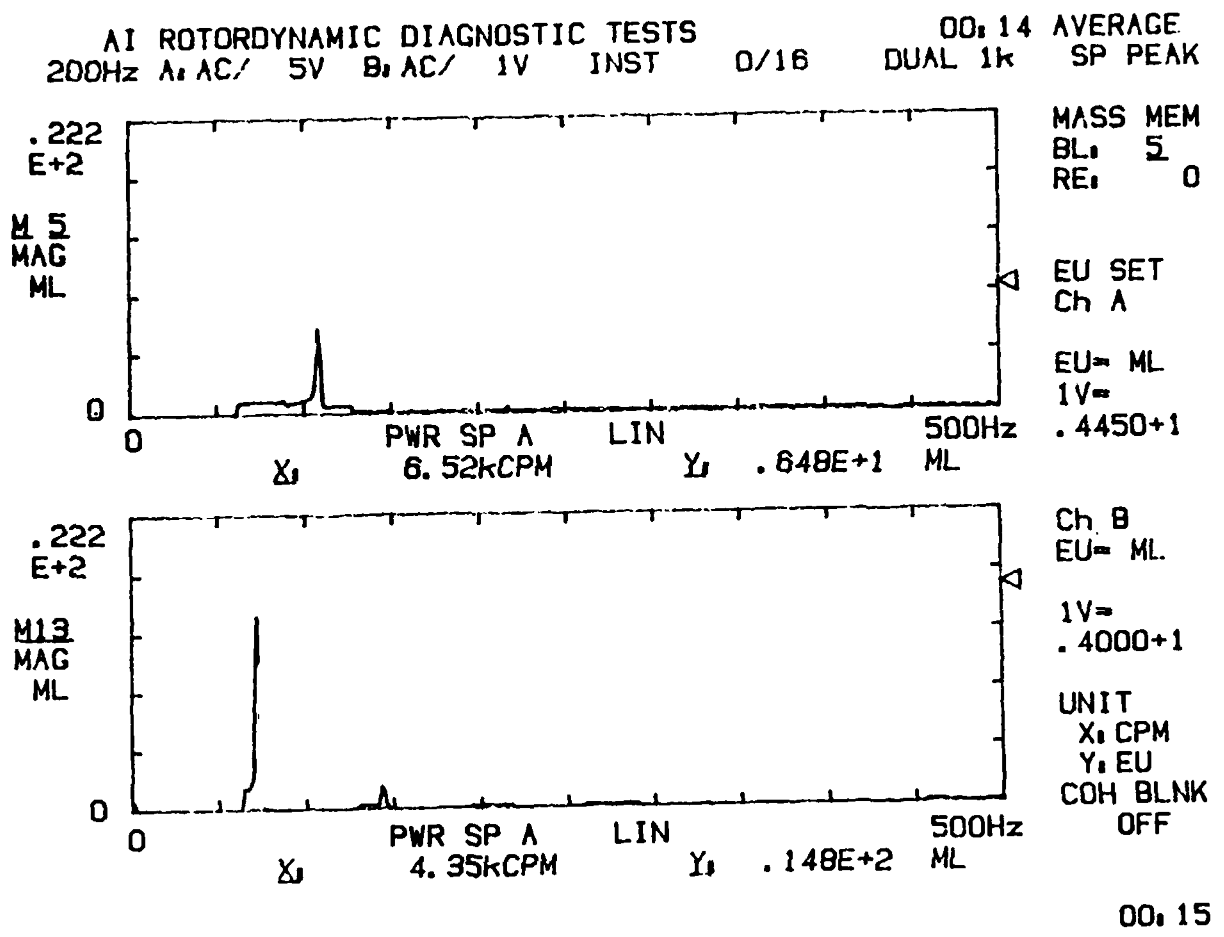


Figure B-8

Table B-5

TABLE VIB-A FOR INCREASED SUPPORT FLEXIBILITY

```
INCREASED FLEXIBILITY - PROBE 2
INTERPOLATION NONE
LABEL   FO      AO       F1      A1      F2      A2     F3      A3

PK1     0.    0.000    4350.  14.800  13050.   .273    0.      0.
PK2     0.    0.000       0.   0.000      0.  0.000    0.      0.
PK3     0.    0.000       0.   0.000      0.  0.000    0.      0.
PK4     0.    0.000       0.   0.000      0.  0.000    0.      0.
PK5     0.    0.000       0.   0.000      0.  0.000    0.      0.
```

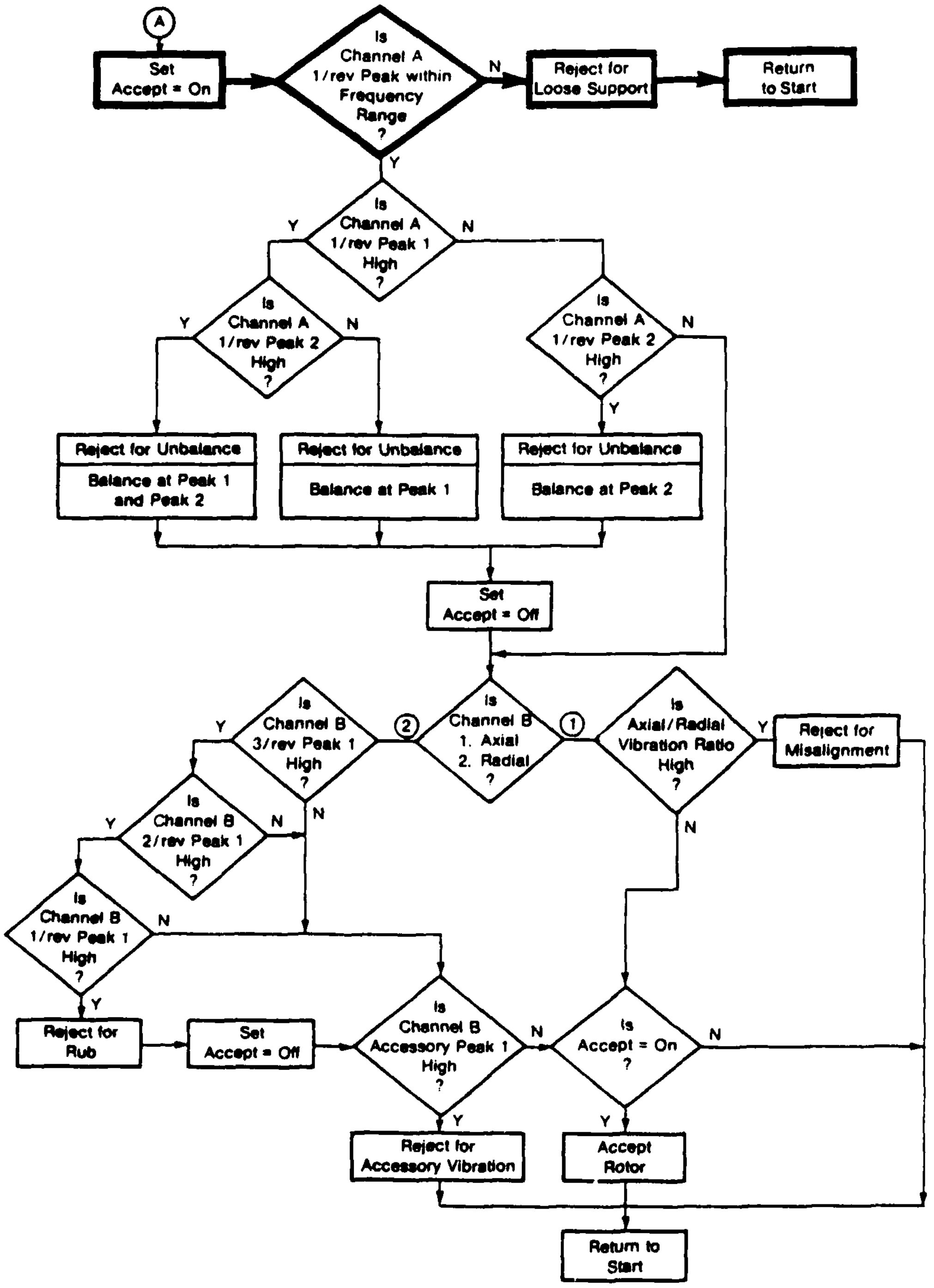

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Axial/Radial Vibration Ratio High ?
Y
Reject for Misalignment
N
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-9

Path Through the Fault Tree for the Increased Support Flexibility Fault

86687-1e

B.6 RUB

The rub fault was introduced by installing the rub bolt with a small clearance between it and the No. 4 disk. When the disk orbit exceeds the clearance, there is rubbing between it and the bolt. If the clearance is about the same as the maximum response at the No. 4 disk (e.g., at the critical speed) then there is only a momentary rub which ceases after either the bolt wears or the orbit diminishes at higher or lower speeds. Thus, to ensure that a sustained rub was induced, the disk/bolt clearance was set at approximately 1.5 mils and the rotor was unbalanced so that the maximum disk orbit exceeded the clearance.

When rubbing occurs between the rotor and a stationary component, there is typically an increase in the amplitude of higher harmonics (i.e., two per rev, three per rev, etc.). The specific harmonics, and the amount by which they increase, depends on the type and severity of rub. Both experience with and analysis of this rig show that for the rub introduced, there is a significant increase in two-per-rev and a small increase in the three-per-rev response. Figure B-10 shows data for Probe 4 for both the baseline case and for the rub fault. In the baseline case (upper trace), the primary response is the one-per-rev at the first critical, and there is no two- or three-per-rev response above the noise limit. The lower trace shows response for the rub event. Note that at the first critical, the one-per-rev response levels off at approximately 1.4 mils (which corresponds to the disk/bolt clearance set at 1.5 mils). Furthermore, two-per-rev response is significant (about two times run out) and there is evidence of three-per-rev response. The presence of these components and their amplitudes are detected by the ES to identify the rub fault. Table VIB-B for the rub fault is shown in Table B-6 and the path through the fault tree is shown in Figure B-11.

B.7 ACCESSORY VIBRATION

The accessory fault was introduced by using an external excitation system on the bracket which supports the No. 4 probe. Due to the data acquisition approach used, it was necessary to have the accessory vibration fault at a frequency not included in the frequency windows used for one-, two- and three-per-rev amplitudes. Furthermore, to get sufficient force out of the excitation system, it was necessary to operate it at the lower end of its 0-1000 hz range. Thus, it

BASELINE
(Probe 4)

RUB
(Probe 4)

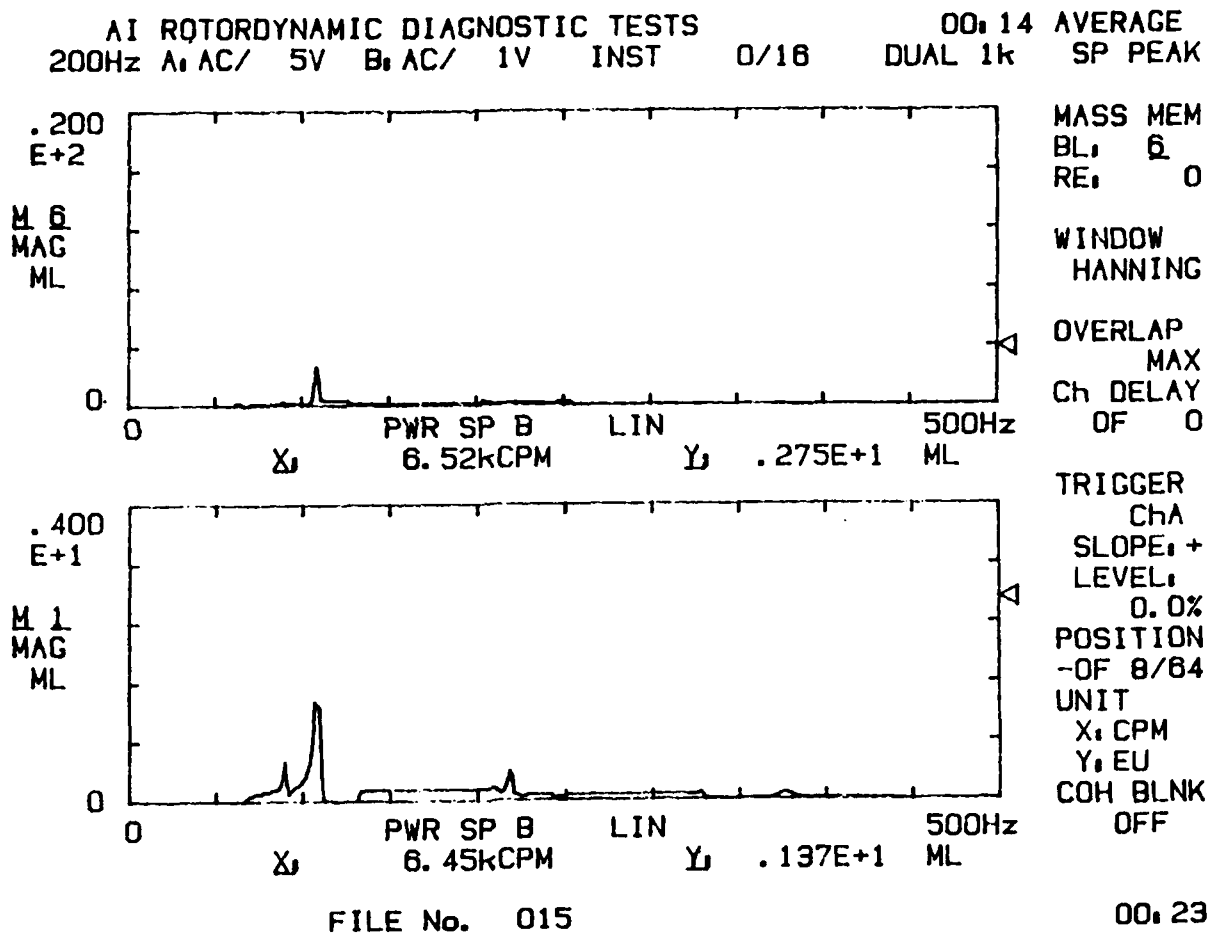


Figure B-10

Table B-6

TABLE VIB-B FOR RUB

RUB - PROBE 4
INTERPOLATION NONE

LABEL	F0	A0	F1	A1	F2	A2	F3	A3
PK1	0.	0.000	6450.	2.050	13050.	.568	1965.0	.090
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.

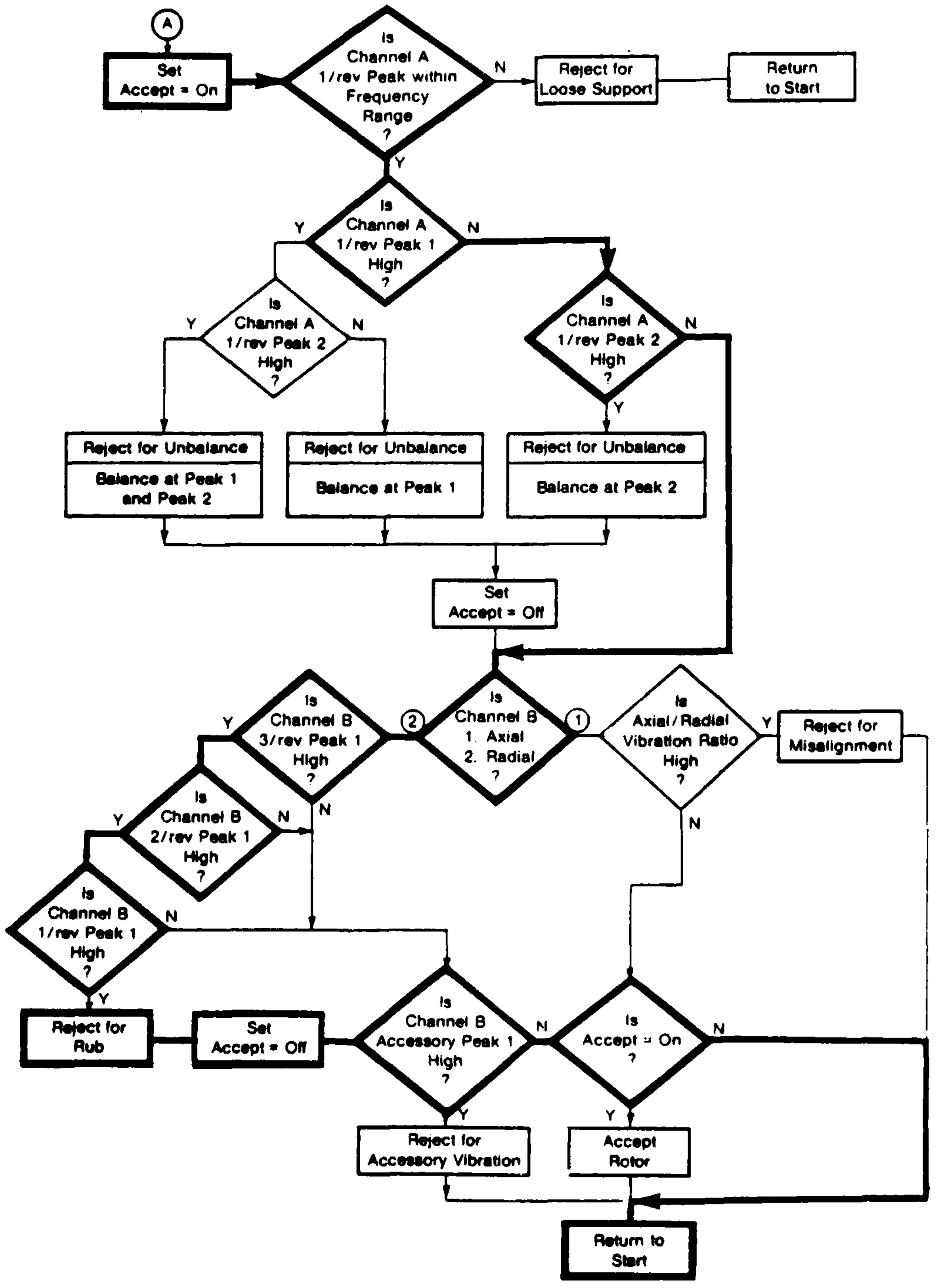

A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Axial/Radial Vibration Ratio High ?
Y
N
Reject for Misalignment
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-11 Path Through the Fault Tree for the Rub Fault was decided to impose accessory vibration in range 0-66.7 hz, and a frequency of 52.5 hz was chosen.

Rig response for baseline and accessory vibration is shown in Figure B-12. For the case with accessory vibration (lower trace), the component at 3,150 r/min (52.5 hz) has an amplitude of approximately 1 mil. The accessory vibration is detected by the ES when this vibration component is present. Table VIB-B for accessory vibrations is shown in Table B-7 and the path through the fault tree is shown in Figure B-13.

BASELINE
(Probe 4)

ACCESSORY
(Probe 4)

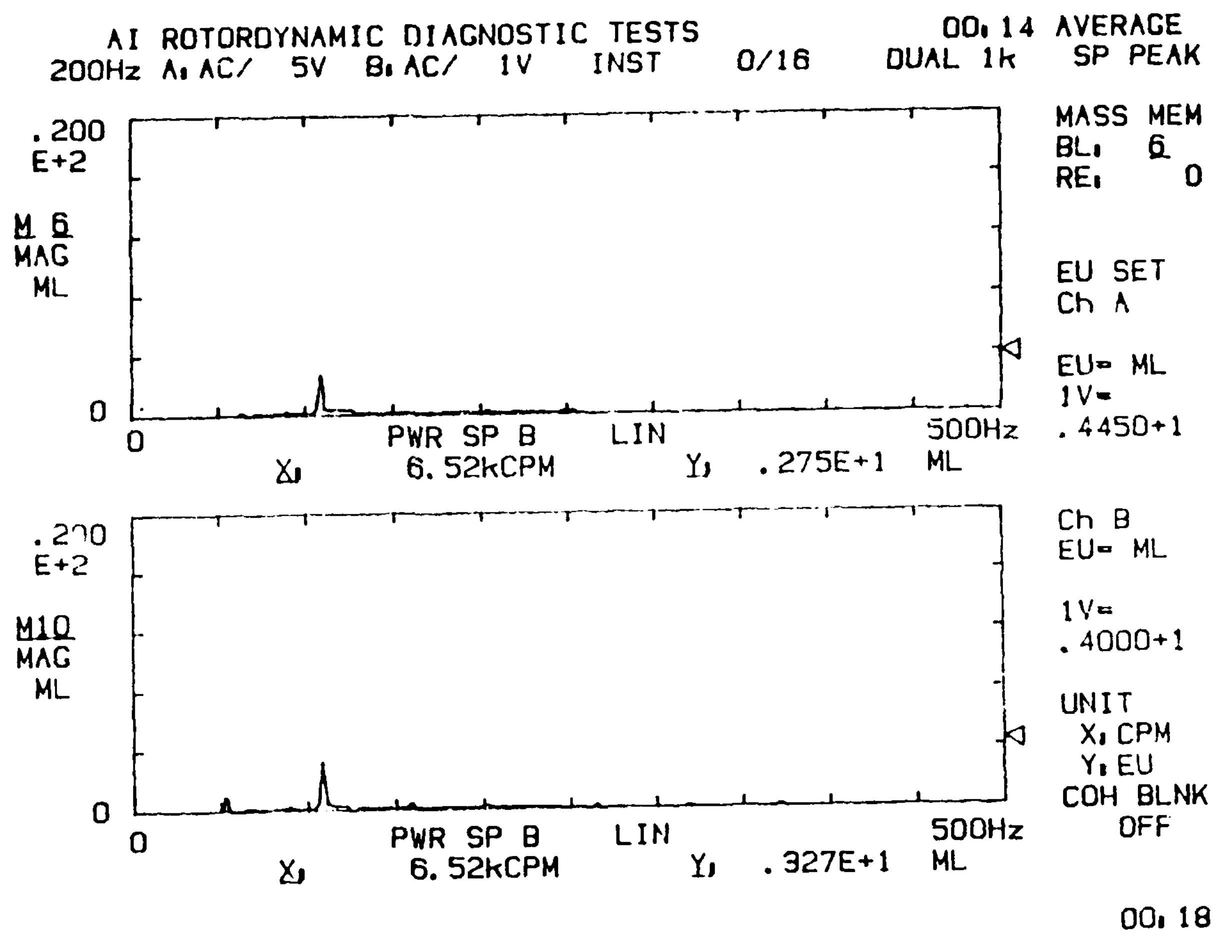


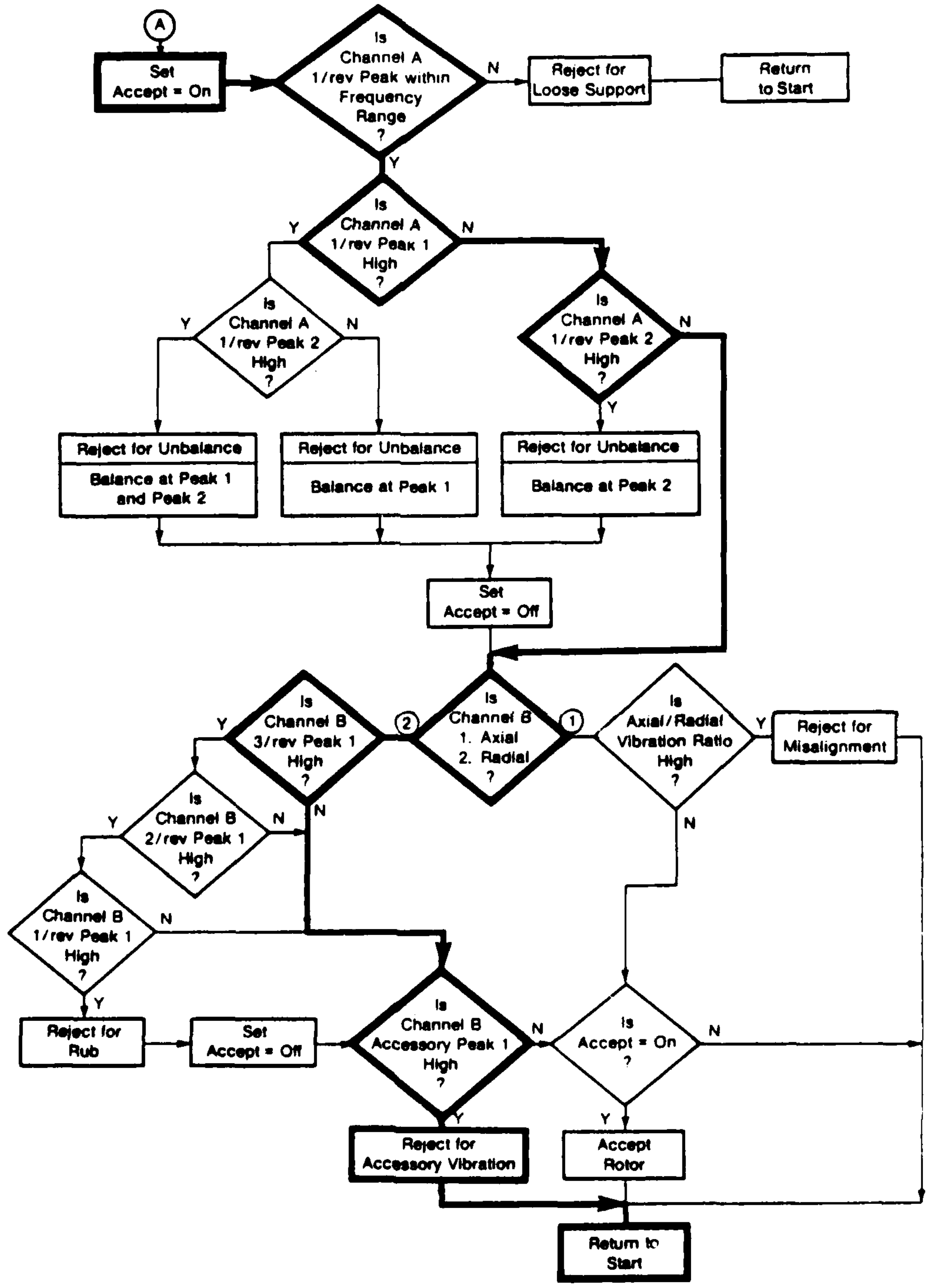
Figure B-12

Table B-7

TABLE VIB-B FOR ACCESSORY VIBRATIONS

ACCESSORY VIBRATION - PROBE 4
INTERPOLATION NONE

LABEL	F0	A0	F1	A1	F2	A2	F3	A3
PK1	3150.	.963	6520.	3.270	0.	0.000	0.	0.
PK2	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK3	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK4	0.	0.000	0.	0.000	0.	0.000	0.	0.
PK5	0.	0.000	0.	0.000	0.	0.000	0.	0.



A
Set Accept = On
Is Channel A 1/rev Peak within Frequency Range ?
N
Reject for Loose Support
Return to Start
Y
Is Channel A 1/rev Peak 1 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
Y
N
Is Channel A 1/rev Peak 2 High ?
N
Y
Reject for Unbalance
Balance at Peak 1 and Peak 2
Reject for Unbalance
Balance at Peak 1
Reject for Unbalance
Balance at Peak 2
Set Accept = Off
Is Channel B 1. Axial 2. Radial ?
2
1
Is Channel B 3/rev Peak 1 High ?
Y
N
Is Axial/Radial Vibration Ratio High ?
Y
N
Reject for Misalignment
Is Channel B 2/rev Peak 1 High ?
Y
N
Is Channel B 1/rev Peak 1 High ?
N
Y
Reject for Rub
Set Accept = Off
Is Channel B Accessory Peak 1 High ?
N
Y
Is Accept = On ?
N
Y
Reject for Accessory Vibration
Accept Rotor
Return to Start

Figure B-13

Path Through the Fault Tree for the Accessory Vibrations Fault

86687-1b

APPENDIX C

EXISTING EXPERT SYSTEMS

C.1 INTRODUCTION

Practical applications for expert systems arose to solve a variety of problems in specialized areas in which there were shortages of human experts. An expert system can capture the practical experiential knowledge that is hard to pin down and that is rarely found in textbooks. The major application areas of expertise that have developed include medicine, oil exploration, equipment repair, education, organic chemistry and mathematics.

DENDRAL, a system which analyzes chemical experiment data to infer the plausible structures of unknown compounds, surpasses all humans at its task. MACSYMA also surpasses most human experts in performing differential and integral calculus. In the medical field, MYCIN was developed to diagnose and treat infectious blood diseases, CADUCEUS consists of a network of relationships between diseases and symptoms in internal medicine, VM is a ventilator manager for monitoring cardiac surgery patients, and CASNET is used for consultations in the diagnosis of glaucoma. (EXPERT is an expert-system-building language that evolved from CASNET, and has been used for building models in ophthalmology, endocrinology, and rheumatology.) Other existing systems include PROSPECTOR, which gives expert advice on finding ore deposits from geological data, ACE which is used for telephone company repair diagnostics, DELTA, a system which diagnoses malfunctioning locomotives and PICON, an intelligent real-time refinery control system. PLANT/DS was developed for diagnosing diseases of soybean plants. In the computer field, DOC identifies the causes of computer crashes, and R1 conFigures computer systems. In the field of education, intelligent tutoring systems have been developed such as SOPHIE, which acts an an electronics laboratory instructor, and SCHOLAR which teaches South American geography. Some of these expert systems are reviewed below.

C.2 EXISTING EXPERT SYSTEMS

C.2.1 DELTA/CATS-1

DELTA (Diesel-Electric Locomotive Troubleshooting Aid), or CATS-1 (Computer Aided Troubleshooting System, Version One), as it has recently been renamed, is an expert system developed at General Electric Company in Schenectady, New York to assist railroads with locomotive maintenance. DELTA was originally developed in LISP, but has been converted to FORTH to allow for successful implementation of the system on a microprocessor. DELTA incorporates approximately 530 rules, 330 of which are diagnostic in nature, and 200 rules are to support the help facility. The DELTA control system combines forward and backward reasoning. In addition to the forward and backward chainers, its architecture consists of the Fact base, which stores all known facts, the Knowledge base, which stores the diagnosis or help rules, the Meta-rule base, which stores meta-rules about the system itself, and the Hypothesis-space, which holds the current diagnostic hypotheses. After case data has been input, the operation of DELTA is as follows:

- The Meta-rule base selects a set of hypotheses from all known hypotheses based upon known facts. It then eliminates all unapplicable rules from the Knowledge base and passes control to the backward chainer. This step only occurs once; it is never reinvoked.
- The backward chainer continues until it reaches a point where it finds a fact that cannot be proven with the current data. The user is then asked to add a new fact, and control passes to the forward chainer. In the event that there is a failure to add new facts at this point, another hypothesis is selected.
- The forward chainer is only invoked by interaction with the user, at which time it adds the new fact as well as all consequences of the new fact to the Fact base. Control is returned to the backward-chainer, and the process continues.

Knowledge in DELTA is represented in the form of a predicate function which is applied to an object, attribute, value triple with a measure of certainty. Predicates consist of measures of equality and confirmation.

The DELTA system not only leads the user through a required repair sequence, but can actually present computer-aided drawings of parts and subsystems in question. DELTA/CATS-1 can identify the cause of a malfunction and present specific repair instructions. The system is joined to a video disk to educate the user as to the repair procedure to follow.

C.2.2 DENDRAL

DENDRAL, one of the oldest, largest and most successful expert systems in existence, identifies the molecular structure of organic compounds from mass spectral and nuclear magnetic response data. It reasons by linking into long chains the empirical rules that expert chemists use to interpret instrument data. This process, called "forward chaining", enables the program to steer efficiently through an ever-branching tree of possibilities without being buried by a combinatorial explosion.

Heuristic DENDRAL uses a strategy called plan-generate-test, in which a planning process that uses constraint-satisfaction techniques creates lists of recommended and contraindicated substructures. The generate-and-test procedure then uses those lists so that it can explore only a fairly limited set of structures. This is not necessarily the same method used by chemists, but is easily understood by them.

DENDRAL has three functional parts, namely, PLAN, GENERATE, and TEST:

- PLAN - Planning in DENDRAL means redefining the problem in terms that reduce the effort of the problem solver. Instrument data is analyzed and constraints derived from mass spectrometry are automatically inferred. For example, it may be inferred that the unknown molecule is probably a ketone but definitely not a methylketone. The output from the DENDRAL Planner is a two-part list of structure descriptions: the GOODLIST, molecular fragments that must be in the final molecular structure, and the BADLIST, fragments that must not appear in the final structure.
- GENERATE - The heart of a heuristic search program is a generator of the search space. The number of chemical graphs for molecular formulas of interest to chemists can be extremely large. Thus it is essential to constrain structure generation to only plausible molecular structures.

CONGEN (Constrained Generator) is a program that replaced the old DENDRAL generator of acyclic structures, and is the hypothesis generator now in use. Unlike the original Heuristic DENDRAL program, it does not infer constraints from mass spectra, but allows the chemist to specify them, thus replacing the planning part of the program. The generation segment allows interaction at every stage so that, with partial results, the chemist may be reminded of additional information that can be specified, thus limiting further the number of possible structures.

- TEST - This last part of the program ranks the resulting list of candidate structures by simulating their behavior in a mass spectrometer. Predicted data is compared to the data from the unknown compound to eliminate some candidates and rank the others.

DENDRAL is structured to read its task-specific knowledge from tables of production rules and to execute the rules in new situations, however domain-specific rules from experts for use by DENDRAL are difficult to extract. In an attempt to find a more efficient means of transferring knowledge into the program, Meta-DENDRAL was developed to aid in building the knowledge base. Meta-DENDRAL is designed to infer rules from empirical data by scanning the data, searching the space of rules for possible explanations and then modifying its rules based on detailed testing. This program has discovered new rules for structures for which rules had not previously been reported.

Rules in the DENDRAL system reason with certainty but deliberately overlook small statistical errors between the rules and the actual empirical data. Knowledge is represented as a special piece of code for the molecular structure generator and as production rules for the data-driven component and evaluator, therefore, knowledge acquisition requires reprogramming or production-rule editing. DENDRAL's speed in producing molecular structures has been at the expense of an ability to explain how a result was derived. The generator of chemical-structure candidates employs a procedure for exhaustively producing possible structures based on various graph-theoretic notions that chemists who use the system are unlikely to know or care about. Thus, the expertise resides in a procedure that is conceptually opaque to the typical user.

Some of the features which have been added to DENDRAL to make it easier to use include graphical drawings of chemical structures, an easily understood language for expressing and editing chemical constraints, on-line help facilities, depth-first problem solving to produce some solutions quickly, and estimators of problem size. The user may question the system at any time as to the amount of work remaining.

C.2.3 DOC

DOC is an expert system begun in 1983 to identify the causes of computer crashes for Prime. It was originally developed in PROLOG with some use of LISP, but is now implemented exclusively in PROLOG. DOC consists of approximately 300 rules, meta knowledge about itself, a backward chaining control mechanism, and a pretty printer for displaying the contents of registers at the time of a crash. A fact was originally represented as a 3-place logical predicate (Prolog) consisting of an object, a value and a weight, the weight being analagous to a certainty factor with a value between one and ten. DOC has since abandoned explicit inexact reasoning, and no longer uses a certainty factor.

DOC has two different styles of operation. First, the data tape produced at the time of the crash is examined, and based upon this data, a top level script is invoked to select a hypothesis. DOC then backward chains to a script which sequentially states the actions to be performed for this hypothesis. During the backward chaining sequence, DOC looks for ALL rules whose consequent matches the object.

Before DOC was developed, the average time for a human to identify the cause of a computer crash was approximately two hours. DOC has saved Prime a considerable number of manhours by reducing this time to about fifteen minutes.

C.2.4 MACSYMA

One of the earliest expert systems to be developed was MACSYMA, which is a large, interactive computer program designed to assist mathematicians, scientists, and engineers in the solution of a variety of mathematical problems, such as algebraic simplification and integration. MACSYMA was implemented based on the premise that a large amount of built-in knowledge is necessary to produce a high-performance program for general mathematics. MACSYMA was originally composed of a set of fairly unstructured LISP functions and now consists of about 230,000 words of compiled LISP code and an equal amount of code written in the MACSYMA programming language. Each MACSYMA rule expresses one way to transform an expression into an equivalent; the solution to the problem requires finding a chain of rules that transforms the original expression into one that is suitably simplified. Most of the algorithms incorporated into MACSYMA were known to mathematicians prior to its development, however a substantial number came about as a result of a research goal to invent and analyze new mathematical algorithms and to extend previously known numerical algorithms to symbolic manipulation.

Users are provided with a variety of expression-transformation commands and a simplifier to automatically apply a set of mathematical rules to every new expression as it is constructed. This is because of the fact that symbolic algorithms often generate large expressions that must be simplified into smaller, more meaningful forms. A semantic pattern matcher is used when applying simplification rules to find instances of a rule's pattern. The user may also define new rules by specifying arbitrary conditions on the pattern variables. A new pattern matcher has recently been developed which will allow the user to specify when inverse axioms are to be used. MACSYMA provides a search-oriented simplifier called SCSIMP to retain the smallest possible form for an expression, however since it is a hill-climbing algorithm, its results are not always guaranteed to be the smallest. MACSYMA maintains a relational database of facts about symbols, stored in the form of a semantic network, in order for the user to specify properties of symbols such as ranges.

MACSYMA, being a large, knowledge-based system, offers the advantage that the user is not required to communicate a lot of mathematical knowledge to the system. However, a disadvantage is that MACSYMA uses complex algorithms which make it more difficult to use and to understand. Many of the algorithms, such as the Risch algorithm for evaluating various types of integrals, while being mathematically correct, are rarely used by human mathematicians because of their complexity. For this reason, much of MACSYMA remains opaque to the user.

MACSYMA cannot reason from basic principles, and thus has limited breadth. There are no explanation facilities and no reasoning under uncertainty. The current system can only be extended or augmented by reprogramming or adding specialized routines.

To assist users with MACSYMA, many on-line aids have been developed, including a frame-oriented interactive primer, an information network, and a program for searching the reference manual. ADVISOR, an experimental version of an automated consultant for novice MACSYMA users, accepts a description of a difficulty from its user and then tries to reconstruct the user's plan for solving his (or her) problem. In addition to these features, a new representation for algebraic expressions using data abstractions and a knowledge-based, plan-based mathematician's "apprentice" are currently under development.

C.2.5 MYCIN

MYCIN, another of the early successful expert systems, is a consultation system to diagnose infectious diseases and suggest therapies. MYCIN has no understanding of the basic medical principles involved in its decisions. It was designed to represent and explain the reasoning process of the system in a manner that is understandable to the knowledgeable user, and also to provide literature sources relevant to the rules that are used in its processes.

The solution method used by MYCIN is backward-chaining from diagnostic hypotheses to data, under the guidance of inferential rules, resulting in an exhaustive depth-first search of an AND/OR goal tree. Knowledge is represented as production rules and knowledge acquisition involves adding new rules to the system. Each fact is a list consisting of a context, a parameter, a value and a certainty factor.

Reasoning under uncertainty is a crucial part of the MYCIN system. Inexact inferences are made based upon confidence or certainty factors ranging between -1.0 and 1.0 that are associated with each fact. These certainty factors are manipulated by the "Calculus of Belief" with results of less than 0.2 of certainty being ignored. Since the rules are inexact and lead to conclusions of less than total certainty, MYCIN considers all the possibilities every time and continues to collect evidence from all other applicable rules even if one rule succeeds. When the system is unable to deduce an answer, it asks the user for the value of a subgoal. To prevent searching for the value of a subgoal when the user might already know the answer with certainty, some attributes have been labeled as "laboratory data", and in these cases, the deduce-then-ask procedure is reversed, and MYCIN attempts to deduce the answer only if the user cannot supply it.

MYCIN examines evidence for and against approximately 100 diagnoses. The final output is a list of possible diagnoses above a suitable threshold of certainty. Because physicians have the final responsibility for treatment, transparency in the system is essential. By typing RULE, the user obtains a human-language explanation of the last rule executed. Typing WHY triggers a back-trace of the inference process that fired the system's last question. The number of the last rule is also given in case the user wished to retrieve it. Repeated use of these commands activates further back-tracing.

MYCIN incorporates good human engineering in a convenient user interface. TEIRESIAS, an auxiliary system built to provide improved explanation capabilities, provides flexible knowledge acquisition and better user-interface facilities. The expert is allowed to inspect faulty reasoning chains and then, through a "nearly natural language," add and modify any rules or clinical parameters required to augment and repair the medical knowledge. Other special features of MYCIN include tabular presentation of medical data and validation experiments conducted for the system.

C.2.6 PROSPECTOR

PROSPECTOR is an expert system designed to aid in mineral exploration. PROSPECTOR employs various kinds of networks to represent knowledge. Rule networks express judgmental knowledge, semantic networks are used for expressing the meaning of the propositions employed in the rules, and taxonomic networks represent static knowledge about the relations among terms in the domain. The nodes in the network correspond to assertions and most of the arcs in the network define inference rules that specify how the probability of one assertion affects the probability of another assertion.

Reasoning under uncertainty is an integral part of PROSPECTOR, with each rule containing two confidence estimates, and a scoring function used to combine the weights. The first confidence estimate indicates the extent to which the presence of the evidence in the condition part of the rule suggests the validity of the rule's conclusion. The second confidence estimate measures the extent to which the evidence is necessary to the validity of the conclusion, or stated another way, the extent to which the lack of evidence indicates that the conclusion is not valid. For example, a PROSPECTOR rule is expressed as:

If: magnetite or pyrite in disseminated or veinlet form is
   present

then: (2, -4) there is favorable mineralization and texture
         for the propylitic stage.

In this rule, the number 2 indicates that the presence of the evidence is mildly encouraging and the number -4 indicates that the absence of the evidence is strongly discouraging for the conclusion. When PROSPECTOR asks yes-or-no questions, the user must indicate certainty about the answer on a scale from -5 to +5, where -5 is a certain no, 0 means don't know, and +5 is a certain yes. Certainty is also required for numerical input.

PROSPECTOR also uses contextual constraints. These work by telling PROSPECTOR, in effect, "Don't even consider hypothesis A unless the likelihood of hypothesis B falls within such and such a range." The range is called the context interval, and is defined by two numbers on the -5 to +5 scale described earlier. The context interval (2,5), for example, indicates that PROSPECTOR shouldn't consider hypothesis A unless confidence in the truth of B is somewhere between lukewarm (+2) and absolutely certain (+5). The default context interval, in effect when no interval is shown, is (0,5).

PROSPECTOR uses a mixed-initiative control mechanism which enables the user to either let the system use a backward-chaining strategy to gather information, or to interrupt the program to select different goals or to volunteer relevant information using key words. The volunteered information is currently limited to simple statements in constrained English which are parsed by LIFER, a natural-language interface facility, and which are represented as partitioned semantic networks.

A model design system, KAS (Knowledge Acquisition System), gives powerful assistance in the time-consuming task of developing the knowledge base used by PROSPECTOR. KAS prompts the user until all missing parts of a new structure are filled in. The core of KAS is a network editor with basic operations allowing it to create, modify, or delete various kinds of nodes and arcs. The network editor knows about the mechanisms employed by PROSPECTOR, protects the user against certain kinds of syntactic errors, and includes a system that keeps track of partial constructs that remain to be completed. At any time, during the operation of PROSPECTOR, the user can turn control over to KAS, and KAS will systematically continue to question the user to fill in the missing parts of the structures. A semantic network matcher gives the user a limited ability to edit by content rather than by form. This process is driven by an external grammar that can be changed without difficulty, making it easy to modify KAS as PROSPECTOR evolves.

The PROSPECTOR group has also developed models that process map data. In these instances, the data for each zone is analyzed separately by the model and a favorability map is constructed which corresponds to the internal confidence measures accumulated by the scoring function. The map is shaded to indicate the degree of favorability of finding the indicated mineral.

C.2.7 R1/XCON

R1 is an expert system which configures VAX systems for Digital Equipment Corporation. It was originally implemented in OPS5 at Carnegie Mellon University, was later renamed XCON by DEC, and is currently used for configuring 97% of the VAX systems sold. R1 consisted of approximately 700 rules; XCON currently has over 3000 rules and is still growing. R1 is considered a dynamic classification model since its intermediate results are highly changeable. During its operation, intermediate decisions are made; i.e. actions are taken, and the model is reevaluated to see what the next decision should be. In a typical classification model, the situation is more stable: new information is gathered, questions are answered, but few direct actions are taken by the system other than synthesizing information supplied by the user.

R1's input is a customer's order for computer equipment and its output is a set of diagrams depicting the spatial relationships among the components as well as an indication of any components missing from the order but needed to complete the system. Although the number of components and design alternatives may be very large, only certain components may be attached to each other. R1 chooses among alternatives based upon these constraints.

Information about components is stored in a database as object, attribute, value triples, with each component having the attributes type and class. Active components are objects in working memory, and are represented as tokens with attribute, value pairs. The working memory also includes information about contexts (subtasks currently being solved), partial configurations and results of computations.

Problems are solved using pattern matching in the working memory of OPS5. The production rule interpreter matches the productions against a subset of information held in the data memory, selects one or more matching productions, executes the selected production, and revises the production (working) and data memories. No confidence measures are used by R1, however since several production rules may be satisfied simultaneously, it does rely partially on the conflict resolution strategy of OPS5.

The operation cycle of R1 basically consists of Match, Conflict Resolution, and Action segments. Initially, the working memory is empty. During the Match segment, every legal instance of every production is found. In conflict resolution, a single production is allowed to execute from the set of legal instances found during Match. A hierarchy of subtasks is formed which are executed during the Action stage. The actions in a subtask are highly variable and the order of solving these subtasks is not very important. Gradually, partial configurations are formed, with the chaining always proceeding in a forward direction. The basic strategy of R1 is that it does all it can within a context before leaving that context. It chooses between rules by selecting the one obtained with the most information. Several rules determine when a new subtask should be initiated, and at this time a new context symbol with a time tag is added to working memory. Each rule has two conditional elements sensitive to context symbols, and each context has an associated rule which deactivates it.

R1 is an example of a system that uses a fixed abstract solution. In general, it does very little search, however the Match mechanism is insufficient for the complete task to be done. The subtask of placing modules on the UNIBUS is formulated as a search for the optimal sequence that fits within spatial and power constraints, and is handled by a different method.

C.2.8 AESOP

AESOP, (An Expert System Engine Operative with Probabilities), is a rule-based inference engine which asks questions about, makes decisions from, and determines the consequences implied by the knowledge built into the knowledge base. AESOP is written in LISP and currently executes on DEC-VAX and IBM PC/XT computers. It is a backward-chaining problem solver, working from hypotheses to facts. A file of hypotheses (potential solutions) and rules (guidelines to finding the correct solution) along with probabilities is set up by the user. When AESOP is invoked, the rule file is loaded, and the user is asked questions drawn from the rules. AESOP then makes decisions about a particular situation by using the rules and the answers to the questions.

One of the important features of AESOP is that questions may be answered with a yes or a no...or a probability factor ranging from 0 to 10, depending upon how certain the user is about a particular piece of information. The user can also respond with a "maybe" (5), "probably" or "likely" (7), "not-likely" (3), or "don't know" (dk). If the user responds "dk", AESOP checks the knowledge base to determine if rules exist that deal with this uncertainty. The "dk" capability is very powerful, since it allows several levels of rules to exist, with each level containing more specific information to help decide the appropriate response to a higher level question.

Since AESOP does not jump backwards, rules must be structured in a top-down manner with those that resolve a "dk" response located below the original rules. If there are no rules for resolving a "dk" answer, the default response is "no". However, the user is given an opportunity to override this default.

AESOP has an "explanation" feature to allow the user to examine its reasoning process. When an hypothesis is awarded 90% probability or more (maximum value is defined by the user in the rules file), it is "confirmed" and printed to the screen with a message to this effect. If all rules have been exhausted and no hypothesis has accrued the maximum probability, the message "no hypothesis can be confirmed with certainty" is printed, along with the current status of all hypotheses to allow you to examine how far each one actually progressed. The user may also, during the course of AESOP's operation, interrogate the system as to why a particular question is being asked.

The knowledge base (rule file) contains those rules to be used for decision making, the hypotheses being investigated, a list of mutually exclusive rules, and detailed information about specific rules to aid in resolving "dk" responses. The probability or confidence factors originally assigned to each rule by the expert is used together with the confidence level expressed by the user as to how well the rule applies to the current problem. As each question is answered, the probability factor is recalculated by subtracting the previous probability from 1.0, multiplying that value by the product of the initially defined probability and the user answer, and then adding that value to the previous probability. For example, if a rule was initially defined as having a probability of 3, the user's answer had a value of 4, and the previous probability had been calculated as .754, the current probability would be:

```
(+ .754 (* (-1.0   .754)  (*  .4  .3)))
```

(NOTE: All defined probabilities and user responses are internally divided by 10.)

AESOP operates by setting up a tree-like structure of "frames." A frame is a list of properties about an entity, similar to relations and attributes in a relational database management system. Each rule is given a unique frame name and a property list containing its fact (the actual text of the rule), its certainty, a list of frames which this rule uses, a list of frames to use next, a list of frames which were used, a list of counters and a list of frames that it competes with. For example: If the user file contains a rule of the form:

```
((we should automate * )
  (or
   (7.0 * requires perceptual abilities outside the
        range of human limits)
   (5.0 it is economically feasible to automate *)
   (3.0 humans don't like to do *)
```

```
          (1.0 * requires exerting large amounts of force
               smoothly)))
```

then, the property list would look like:

```
      (FRAME1 (FACT (VALUE ((WE SHOULD AUTOMATE *))))
      (CERTAINTY  (VALUE (0))) (USE (USES ((FRAME2 7.0))
      ((FRAME3 5.0)) ((FRAME4 3.0)) (FRAME5 1.0)) (NIL))
      (USED (NIL)) (COUNT (VALUE (0))) (COMPETES-WITH (VALUE (FRAME6))))
```

To minimize the number of redundant questions that the user is asked, AESOP includes "remember" and "recall" functions to store and recall user responses. There is also an "opposites" feature which prevents questions to the user about rules which are direct opposites of each other. When a user responds to a question and its opposite is in the knowledge base, the corresponding inverse probability factor is given to its opposite.

AESOP allows for an initial prompt function to be stored in the knowledge base. This function lets the user describe the problem being solved. The description is then substituted for the wildcard (*) in the rules located in the knowledge base. AESOP is available in the public domain and should be portable, with little modification, to any computer running a version of LISP.

C.3 PROGRAMMING LANGUAGES FOR EXPERT SYSTEMS

C.3.1 LISP

The LISP language was developed to enable computers to manipulate symbols. Computers represent everything as a string of binary digits, ones and zeros. Commonly, these binary digits are interpreted as a code for decimal digits. But in general terms, groups of bits can be used to represent anything desired by a proper organization.

In LISP, the fundamental things formed from bits are work-like objects called atoms. Groups of atoms form lists, which in turn can be grouped together to form higher-level lists. Atoms and lists collectively are called symbolic expressions. Working with them is called symbol manipulation or list processing.

LISP gives the programmer a lot more control and presents information in a much more usable form.

Instead of being described as sequences of steps, LISP programs consist of functions defined in a rather mathematical format. Each function call is represented as a list, the value of whose first elements is the name of the function and the values of whose other elements are the arguments.

Thus, the basic LISP procedure specification involves, not a sequence of program steps, but a function definition in terms of applications of functions to arguments.

One characteristic of LISP that is unique among high level programming languages and that seems particularly important in AI work is the representation of the programs themselves in the same data structure as all the other data, namely, list structure. This is a highly useful characteristic in applications where a program has to explain its line of reasoning, program editors and debuggers.

LISP language syntax is so simple, LISP programs that produce other programs as their output do not have to worry about expressing their results in a complex format.

The LISP language is conducive to the development of sophisticated programming tools because it is easy to write programs that manipulate other programs. The core syntax for the LISP language is simple, and LISP programs are naturally represented in simple LISP data structures in a way that reflects the structure of the program. Since LISP requires no declarations, programs can be built up incrementally, thus supporting structured growth style of program building.

The internal representation of a LISP program is the same as that of any other multilevel list. LISP is unique among programming languages in storing its programs as structured data. It is particularly easy to write LISP programs that generate LISP expressions and programs, as in automatic programming. Functions can be passed as parameters to other functions. LISP can be a foundation for more advanced languages by writing an interpreter in LISP for a new LISP-like language. The special AI languages MICRO-PLANNER, CONNIVER, and QLISP were all implemented in this way.

Interpretive execution helps incremental programming once a program is fully operational, it is usually compiled for greater speed.

Interpretation of LISP is far easier than for most languages, because of its uniform syntax. Other features like dynamic allocation and the absence of type declarations also suit it to interactive use.

IQLISP is an interpreter thus allowing interactive program entry and debugging. IQLISP has functions that utilize 8087 math support, function key capturing, display windowing, and many MS-DOS interfaces. IQLISP has floating point numbers in single and double precision. It has the full range of arithmetic, recognition and comparison functions for numeric data. These tools will facilitate mathematically intensive programs such as in calculus, matrix manipulation, and polynomial evaluation.

Character strings and names may be 0 to over 32000 characters long. Strings may be searched, substringed and concatenated. IQLISP manages these large data items efficiently by storing each unique name once, in a garbage collectable area, and use typed pointers to them for assignment, type recognition, and comparison. IQLISP has functions to store, fetch and remove property flags and name-value pairs.

This makes attachment of extra information to literal atoms easy.

IQLISP provides sequential file and display window I/O functions, and allows an arbitrary number of files and windows to be used.

IQLISP has true arrays and array functions for efficient table information.

IQLISP provides a variety of list management functions. Data structures such as arrays, records, stacks, queues, trees, and other LISP functions can be built under program control.

Programs require control structures as well as data structures. LISP control structures are function calls, using choice-making, looping, recursing, and error-trapping functions, and providing assembly language and hardware interfaces.

IQLISP allows users to write their own macros, which will expand into LISP code at execution.

Users functions are recursive, and can be defined to receive either a fixed or variable number of arguements evaluated or unevaluated before being passed in as needed.

IQLISP has error trapping and stack manipulation functions, various system management and garbage collector calls, and interfaces to MS-DOS and DEBUG for assembly language routines.

INTERLISP is a programming environment based on the LISP programming language. INTERLISP has an extensive set of user facilities, including syntax extension, uniform error handling, automatic error correction, an integrated structure-based editor, a sophisticated debugger, a compiler, and a filing system.

INTERLISP has been used to develop and implement a wide variety of large application systems. Examples include the MYCIN System for infectious disease diagnosis, the Boyer-Moore Theorem Prover, and the BBN Speech Understanding System.

Interactive program development consists alternately of testing of program parts and editing to correct errors discovered during the tests and/or extend the program. INTERLISP supports both the testing and editing operation. The user works exclusively with the INTERLISP System during the interactive session. During this process the primary copy of the program resides in the programming system as a data structure. Hence, INTERLISP is called a residential system.

A file package automates the bookkeeping necessary for a large system consisting of many source files and their compiled counterparts. The file package removes from the user the burden of keeping track of where things are and what things have changed. The file package assumed a degree of autonomy and operated automatically and behind the scenes.

MASTERSCOPE, a component of the INTERLISP environment, is an interactive program for analyzing and cross referencing user programs that addresses this problem. It contains facilities for analyzing user programs to determine which functions are called, how and where variables are bound, set or referenced, which functions use particular record declarations, etc.

DWIM (Do What I Mean) facility in INTERLISP is an impressive feature, which is invoked when the basic system defects on error and which attempts to guess what the user might have intended. When an unrecognized file package command, edit command, LISP function, etc., is encountered, the spelling corrector attempts to find the closest match within a list of relevant items. When DWIM corrects a user's misspelled function name in one of his programs, it actually modifies the user's program to contain the correct spelling.

The various forms of the INTERLISP iterative expression permit the user to specify complicated loops in a straight forward and visible manner. An iterative expression consists of a sequence of operators, indicated by keywords, followed by one or more operands; many different operands can be combined in the same iterative statement.

Another tool in the INTERLISP environment is a programmer's assistant which records, in a data structure called the history list, the user's input, a description of the side effects of the operation, and the results of the operation. For example, the UNDO command can be used selectively to flip back and forth between two states. The user might make some changes to his program and/or data structures, run an experiment, undo the changes, rerun the experiment, undo the undo, and so on.

The INTERLISP programming environment is friendly, cooperative and forgiving. It is an integrated system. The various facilities themselves can use each other in important ways, since they all coexist in the same address space. There is no need for context switching.

The INTERLISP environment is extensible via substitution macros, which associate a template with the new command. In addition computed macros are also supported. A computer macro is a LISP expression, evaluated to produce a new list of operators/commands/expressions.

INTERLISP in the personal computer environment has some implications. On the PC, the response of DWIM, although slow, can be interspersed with the time the user takes to think about the problem. It even becomes reasonable to devise tools that operate continually in a background made while the user is thinking, such as an incremental garbage collector, a program that updates the masterscope data base, or one that performs compilations. PC's thus cause a qualitative change in the programming environment, because the machine can be working continually for a single user.

A significant addition to the INTERLISP on the new generation of PC's is the availability and integration of very high-resolution and high band width displays. Use of pointing devices, menus, vivid displays of intermediate results, and windows enhance the capabilities.

C.3.2 PROLOG

PROLOG includes a database of facts and rules, from which it can draw conclusions. A program in PROLOG is a set of facts and rules with the same name and number of arguments stated in a restricted form of logic. PROLOG interpreter is a program that proves theorems. PROLOG is called a "logic programming" language because it is based on a form of logic. What further distinguishes PROLOG is its ability to prove theorems relatively efficiently.

PROLOG programs can be comparatively simple, and they don't need any form of control, such as for loops or go to's. PROLOG will automatically try all possible alternatives until one, if any, meets the constraints expressed in the original query. The key point is that one doesn't have to specify how to find the solution.

PROLOG is very portable. Since it is so machine-independent and easy to implement, specialized hardware is not required for rapid software development.

PROLOG could also be used to interface with such devices as modems. It could not only look up or number and dial it, but also know about alternative numbers.

C.3.3 OPS5

OPS5 is a member of the class of programming languages known as Production Systems. It is used primarily for applications in the areas of artificial intelligence, expert systems, and cognitive psychology.

Three interpreters for OPS5 have been written, one in BLISS, on in MACLISP, and one in FRANZ LISP.

A production system is a program composed entirely of conditional statements called productions. These productions are stored in a memory called production memory. The productions operate on expressions stored in a global data base called working memory whichis separate from the production memory. The production is similar to the if-then statement of conventional programming languages.

Production systems differ from conventional programs in two major respects. The first is that the production system uses a different method for encoding the state of a computation. A conventional program encodes state by assigning values to local and global variables. A production system encodes state by putting expressions in the systems global working memory. A conventional program uses sequential execution of statements plus a number of control constructs including subroutine calls, loops, and conditional branching. A production system uses LHS (left hand side which is the conditional part of a production) satisfaction. Each production's LHS is a description of the states in which the production is applicable; the LHS becomes true when there is some information in working memory that the production can process. When the interpreter performs the match process, it is in effect searching for a production that knows how to process the data that is in working memory. When it finds that production and executes its RHS, working memory is changed, and so on the next cycle, the interpreter performs the match again to find a production that can handle the new data.

OPS5 is unique in having been used extensively to build operational expert systems, among them, R1 EXCON, and AIRPLAN.

OPS5e is an enhancement package for OPS5 and was developed by Verac, Inc. It was designed to bring the full power of the symbolics LISP machine to bear on the use of OPS5 in building expert systems.

The OPS5e package integrates OPS5 into the ZETALISP language and LISP machine window system. This integration takes advantage of the large address space and the incremental garbage collection features of the Symbolics LISP machines. Specific development environment features include:

- window based development and debugging
- an OPS mode for the ZWEI test editor
- improved run-time efficiency
- improved syntax error detection
- improved top-level commands
- on-line help.

A production system written in the OPS5 language consists of three logical components: the production memory (pm), the working memory (wm), and the interpreter.

Production memory holds long-term, expert knowledge in the form of production rules (also termed productions or rules). Each production rule is in the form

IF Left-Hand-Side Pattern (LHS)
THEN Right-Hand-Side Action (RHS)

The working memory holds short-term knowledge in the form of a collection or working memory elements (wme), which evolves in the course of solving the problem at hand.

The interpreter is the inference engine. If finds matches of the LHS of production rules to working memory elements, resolves conflicts where more than one rule matches, and fires the rules, i.e., causes execution of the RHS of the selected rule. The most typical result of firing a rule is to alter the working memory by adding, deleting or modifying one or more elements. This sequence is one cycle of the process. This recognize-act cycle is repeated as long as there is a rule ready to fire.

The philosophy of production system design promulgated by the designers of OPS5 is that the production memory grows large as the knowledge engineering effort is able to capture more and more expertise. The working memory is envisioned as transient and relatively small. And the interpreter imposes no particular control structure on the process.

C.4 AI COMPUTER HARDWARE

The recent breakthroughs in AI development are due in part to the availability of powerful computer systems. Symbolics and Lisp Machine Inc. (LMI) make computers specifically for symbol-manipulation languages such as LISP. The VAX, from Digital Equipment Corporation is also very popular with a number of AI developers. These and other hardware systems provide differing capabilities and are limited in the languages they can use. These capabilities and limitations strongly affect both the choice of computer(s) for system development and the eventual "target system" (for the end user). The following paragraphs describe available computer hardware and hardware-related considerations that must be taken into account when planning the implementation of an expert system for rotor diagnosis.

C.4.1 SYMBOLICS 3600

Symbolics, Inc. was founded to develop, manufacture, sell, and support computers for symbolic computation. Its first product, the LM-2 is based on the Lisp Machine, a revolutionary symbol-processing computer used for interactive software development and computer-aided design. The LISP-based processor was designed and developed at MIT's Artificial Intelligence Laboratory by Symbolic's founders. In 1980 they incorporated Symbolics and licensed this technology from MIT to further develop smybolic computation as the mainstream computer technology of the 80's.

The Symbolics 3600 Symbol Processing System is a single-user, LISP-based computer system that is specifically designed for symbol processing applications such as artificial intelligence. It is a 36-bit, tagged, stack-oriented, computer system that is reputedly substantially more powerful, faster, and cost-effective than many other LISP-based processors. It features high-resolution black and white and color graphics, Ethernet network compatibility, a powerful ZETALISP operating system, plus an extensive systems software library. The 3600 processor is built around a dedicated high-performance microprogrammed 36-bit stack-oriented and memory-tagged architecture with 32-bit datapaths. The instruction fetch unit is overlapped with normal execution and permits many opcodes to execute in as little as 200 nsec. Other characteristics include:

- Stack-oriented, with high-speed buffering of the top stack frames.
- Run-time checking, assisted by hardware, for data-type mismatches, uninitialized variables, and array-bound errors
- Fast array indexing
- IEEE 32-bit floating point
- 256 million 36-bit words (1 giga-byte) virtual address capability
- Up to 7.5 MWords of physical memory.

The 3600 includes an MC68000-based front-end Processor (FEP) that performs two functions: during normal operation the FEP controls low and medium-speed I/O devices and does error logging and recovery; when the 3600 is not running, the FEP is used to diagnose the machine.

Devices such as the keyboard, the mouse, and the serial lines are connected to the 3600 via the FEP. An optional cabinet may be used to attach additional commercially available MULTIBUS or UNIBUS peripherals to the 3600. Programs can be down-loaded into the FEP to control these peripherals. When the 3600 detects errors, the FEP logs them for diagnostic purposes.

For diagnosing the 3600 CPU, the FEP has access to all CPU registers and buses, and to the main memory, disk, and network. For diagnosing microcode, the FEP can single step and breakpoint the CPU. All debugging functions can be accessed from the console of the 3600 through the local network or over a serial line (possibly connected to a modem for remote diagnosis capability).

The 3600 uses disk files as its long-term storage mechanism. It comes with a hierarchical file system that can be accessed locally or remotely over the high-speed local network. A 3600 system with large disks can be used as a dedicated file system resource to serve a community of 3600's. Full backup and archiving to tape are provided. The robust file system can recover from most crashes without running a salvager because disk blocks are written redundantly. The 3600 file system allows users to add their own attribute types.

The 3600 can also access the file systems of time-shared mainframes. Currently, the file systems of DEC VAX computers using VMS or BSD UNIX

The primary language of the 3600 is Symbolics' Zetalisp, in which all system programs are written. Zetalisp is an expressive, efficient and extensible programming language. User applications systems written in Zetalisp merge naturally into the integrated programming environment; no syntactic or semantic distinctions are made between system software and the applications programming language, or environment.

The Symbolics 3600 software system is a complete programming environment with advanced program development tools integrated in a coherent and consistent way. These tools include:

- Advanced window system to manage the bit-map display.
- Full support for optional high-resolution bit-mapped color graphics display
- Mouse-oriented Display Debugger
- Resident incremental compiler fully integrated with the editor
- Remote file access and robust local file system
- Powerful electronic mail facility
- Languages: ZETALISP, FORTRAN 77, PASCAL and INTERLISP compatibility.

Processing power of the 3600 is delivered with less hardware than many super-minicomputers, due to such innovations in CPU architecture as memory-tagging and hardware-supported data typing. Optimized to execute the LISP language, the system's performance competes favorably with other LISP machines.

C 4.2 LISP Machines, Inc. (LMI) Lambda

The major design philosophy of the Lambda machine is the concept of modularity. The system is designed so that it can exist in a large number of configurations centered around a 32-bit high performance bus. The motivation for this philosophy is the desire to move away from the conventional notion that a computer maintains a stagnant design, fulfilling needs for only one specific task. The Lambda machine provides a hardware environment for the customization of a computer system so that Lambda configurations are flexible enough to address many different applications. Configurations can include either a LISP processor, a 68000 based UNIX processor, or both. Memory, network interface, video, and disk are all added as the application dictates simply by plugging the appropriate cards into the bus.

The Lisp processor is available in two cofigurations:

1. <u>32-Bit Configuration -</u> This configuration is identical to that of the present LISP Machine, having a 32-bit word length which contains a 24-bit pointer, five bits of data type and three bits of special storage information. This processor, while operating at considerably higher speed, will run the identical software as the CADR.

2. <u>40-Bit Configuration -</u> An optional configuration for the processor is the 40-bit machine which extends the pointer size and, hence, the addressable virtual memory to 32 bits. The remaining eight bits are unchanged for the existing machine. In this manner, there will be maximum possible software compatibility with the existing design while substantially increasing the amount of addressable memory in the machine. This configuration furthermore expands the maximum integer and floating point number size to 32 bits which has allowed the implementation of the IEEE floating point standard. The hardware is designed so that this functionality increase will not sacrifice machine speed and the 40 bit version will, therefore, run at the same rate as the 32 bit version.

Architectural features of both processors include:

- 4K word high speed cache memory.
- 64K x 64 bit user-writable control memory - used in conjunction with the system microcompiler (translates LISP source directly to system micro-code).
- 16 x 16 hardware matrix multiplier.
- Larger and faster internal register memories.
- Hardware implementations of many microcode intensive tasks.
- Hardware garbage collection assist.

In addition to the LISP processor, the Lambda machine also offers a processor configured with the 10 MHZ version of the Motorola 68000. This processor is configured with a 4K cache and executes the UNIX operating system providing processor speeds of about 1 million inferences per second. 68000 UNIX supports a multi-user/Multi-process environment, virtual memory via demand paging and is based on Berkeley-UNIX.

Built on top of the hardware architecture is a very extensive LISP implementation compatible with MACLISP which provides thousands of very rich and useful features. These features naturally extend the power of the basic LISP language in ways that could not be done on conventional architectures. Additionally, the LISP Machine also provides an extensive array of software development tools. Along with the basic LISP interpreter, the system provides a powerful screen-oriented text editor, compilers that generate either micro or macrocode, as well as a complete user environment with excellent debugging, networking and file system software.

At the heart of the LISP Machine software environment is the ZMACS editor. ZMACS' power derives not only from its inherent text and LISP editing capabilities, but also from its integration into the global LISP environment. Thus, it is both an important system utility and an example of the unified yet modular structure that makes the LISP Machine environment so powerful. The characteristics of ZMACS include:

- Rich command set
- Sophisticated command set
- Integrated graphics
- Graphics oriented error display

- User friendly environment

An important feature of the LISP Machine software environment is its garbage collector. An impediment to the acceptance of LISP as a language for production programming has been its garbage collector. Earlier LISP implementations would periodically pause while garbage collection took place. While an annoyance, earlier LISP programmers accepted moderate delays in order to access the LISP programming environment. With the advent of more sophisticated LISP systems, accessing much larger address spaces, these delays have become a good deal longer and quite unacceptable.

The LISP Machine environment solves this problem by offering a parallel garbage collector. By executing concurrently with other LISP Machine processes, the garbage collector does its job "incrementally", allowing LISP computation to proceed without delay. Users need not fear that a complete garbage collection might occur during a time-critical piece of code. With the addition of a parallel garbage collector, the LISP Machine architecture allows a LISP to directly compete with other "real-time" programming languages, yet still retain its enriched programming environment.

The UNIX software environment for the LAMBDA includes:

- C
- PASCAL
- FORTRAN 77
- Window System
- Several screen editors, including EMACS

One of the more exciting aspects of the Lambda machine is the integration of LISP and UNIX systems. Systems configured with both processors have the ability to communicate on a process-to-process basis utilizing the system defined STREAMS/PIPES interface. Typical applications for this configuration are to utilize the UNIX processor as a multi-user front-end package and to send requests to the LISP processor. LISP would then provide the specific symbol manipulation aspects of the problem solution and return its result to UNIX. This approach would provide an effective multi-user interface to expert system executing in the LISP processor.

C.4.3 Digital Equipment Corporation VAX 11

The Digital Equipment Corporation VAX series computers are 32-bit architecture, super-minicomputers that give applications programs a 4-gigabyte virtual address space. The VAX family of computers ranges in size from the VAX station 100 single-user work station through the VAX 11/725, 11/730, 11/750, 11/780 multiuser systems and finally to the VAX 11/785 dual-processor configuration. All members of the VAX family use the VAX/VMS operating system and feature full transportability of software across all members of the family. The VAX has been used extensively in engineering applications and is a leading contender in the arena of artificial intelligence machines.

The VAX provides the performance, reliability, and programming features often found only in much larger systems. It provides 32-bits addressing for possible memory expansion of up to 16 megabytes and 32-bits arithmetic and data paths for processing speeds and accuracy. The features of the VAX include the following:

- Hardware floating point processor
- Event-driven interupt structure
- Time sliced operation
- Multitasking capability
- Multiuser capability
- Multiprogramming capability
- Interprocess communication and synchronization
- High degree of protection among users
- Operating system capable of simultaneous timesharing, process control and batch processing.
- Programming support for multiple language (FORTRAN 77, PASCAL, PROLOG, LISP, etc.)
- Common language interface that allows procedures written in one language to call procedures in another language.

DIGITAL is firmly committed to the growth of AI usage for the VAX and has available a COMMON-LISP language and expert system development tools such as OPS5. VAX LISP is an implementation of COMMON LISP, a recently defined and extremely important dialect of LISP. The features of VAX LISP include:

- Common LISP compliance
- Rich set of data type and functions
- Integration with VMS common language environment
- Interpreter/Compiler
- Dynamic linking (allows mixture of compiled and interpreted code)
- LISP editor
- Debugging facilities (examine, step, trace, break)

Digital's OPS5 for VAX brings to artificial intelligence a programming language that is specially designed for developing expert systems. OPS5 for VAX is a highly optimized, high performance implementation of the OPS programming language. The highlights of OPS5 for VAX include:

- OPS5 for VAX applications will run like conventional software applications on a VAX system running VMS, so installation is easy and requires no new hardware.
- OPS5 for VAX can call and be called by routines written in any language supporting the VAX Calling Standard, increasing programmer productivity and program capabilities.
- Because OPS5 runs on VAX computers, an expert system can be integrated with VAX real-time applications.
- OPS5 for VAX is nonprocedural, so coded rules can be grouped in any order for convenient maintenance thereby saving programming time.
- The on-line help facility of OPS5 for VAX makes it easy to use the language.

Digitals commitment to the VAX as an AI Machine is demonstrated by the fact that DEC's expert systems XCON and XSEL is implemented on a VAX using VAX, LISP and OPS5 for VAX.

Unlike most AI machines, the VAX has an I/O bus, allowing an expert system to do realtime applications, and the VMS operating system can run multiple, simultaneous processes. While the system accomplishes realtime functions, an OPS5 program can perform expert system functions. The realtime capability would be highly advantageous in an expert system for rotor diagnostics. VAX systems also have the timesharing capability to deliver expert system applications at the lowest cost per user. In addition, The VAX family offers the graphics of the VAX station 100; the communications facilities of DECnet, CI, and Ethernet; the data management software of DATATRIEVE, DBMS, and FMS; and all the industry-standard layered languages.

C.4.4 Xerox 1100

The Xerox 1100 is a dedicated personal computer work station that supports the INTERLISP-D and SMALLTALK-80 programming environments. Both environments take ful' advantage of the personal computer features of the 1100, such as the high-resolution bit-mapped display and the mouse, to facilitate rapid prototyping and to advance system development. INTERLISP-D, a powerful, personal implementation of the INTERLISP dialect of LISP combines the powerful programming features of INTERLISP and the machine features of the 1100. While retaining the facilities that have made INTERLISP widely used in research and development, the XEROX 1100 makes it feasible to deliver INTERLISP-based end-user applications in areas such as interactive symbolic processing, knowledge engineering and expert systems.

C.4.5 Microcomputers

While the above AI machines offer superior performance in a LISP programming environment, their cost sometimes limits their application to the more complex expert systems. Recently, with the advent of implementations of LISP languages such as MU-LISP and IQLISP that run on inexpensive personal computers, such as the IBM PC-XT, small computer systems have become viable candidates for less complex expert systems. Several hardware vendors have introduced micro computer based products aimed at the AI environment.

The tektronix TEK 4404 is an example of such a system and offers features such as:

- Smalltalk-80 programming environment
- Motorola 68010 32-bit micro processor with hardware
- Hardware floating point accelerator
- 1 Mbyte high-speed dynamic RAM memory
- Operating system with multi-tasking and hierarchical file system
- C compiler with a standard I/O library provided

C.4.6 Other Hardware Considerations

- Virtual memory management for 8 Mbyte address space
- 640 x 480 monochrome bit-mapped display, 60 Hz, non-interlaced
- 1024 x 1024 display bit-map with smooth panning
- Three button mouse

Available options include:

- Additional 1MB memory
- Ethernet interface 40 MB hard disk with tape streamer
- Franz LISP programming language
- Prolog programming language
- EMACS Editor

Digital Equipment Corporation's Micro-VAX-I and Micro-VAX-II systems, while not suited as LISP based AI development environments, are extremely good candidates on targets for expert system. The Micro-VAX supports the VAX/VMS environment and permits VAX-LISP applications to be down-loaded for execution. The combination of a central VAX for development and Micro-VAX systems as targets provides a cost competitive approach to the implementation of expert systems.

For the past two years, MTI has been evaluating the capabilities of an IBM PC for AI applications and has successfully demonstrated several expert systems. A rich set of AI developmental tools has been developed for creating, displaying and maintaining expert system knowledge bases. Interfaces to analog-to-digital convectors and graphics input devices have been implemented.

While large-scale expert system software development requires a powerful LISP-based development system, implementation systems are clearly feasible on micro-computers such as the IBM PC and micro-VAX. This prospect is becoming more likely with the trend toward a standardized dialect of LISP known as COMMON-LISP which already has versions that run on the IBM PC. The use of COMMON-LISP is extremely important since portability of software is fundamental to the use of micro-computers for AI applications.

In addition to the LISP machine, two other categories of hardware devices merit special attention in expert systems. These are information input and information output devices tailored especially for use by computer users who either are not highly trained in the use of computers or who are somewhat uncomfortable when faced with the prospect of operating a computer. Since the usefulness and effectiveness of the expert system may be dramatically affected by these users, the simplicity of input and output devices will be key to system development.

Providing alternatives to the standard computer keyboard and printer is one way to implement innovative and anxiety-reducing operator interfaces. Touch-sensitive screens, function buttons, graphics tablets, bar code readers, light pens, and mouse-like graphic pointers are being more widely used as alternatives to the keyboard. With devices like these, the expert system user can interact with the system by pointing to a desired function, touching a symbol that represents the function, drawing a symbol that represents the function or by letting the system read and interpret a special bar code label.

Hardware to implement these approaches is available for almost every computer system. The LISP programming environments (LMI, SYMBOLICS, etc.) heavily use graphics displays and pointing devices presently. Incorporation of support for other of these devices would further enhance the interface to an operator.

By using these and similar techniques, specific input formats can be devised for a particular application that will permit the nonexpert to communicate effectively with the expert system in terms of objects, concepts and ideas that are familiar and comfortable to the user and still meaningful to the computer.

Another approach to alternative methods of computer input includes a group of devices that permit the expert system to automatically input the information that it requires. Analog-to-digital converters, vision subsystems and speech recognition subsystems enable the automatic input of information. Interest is growing in the development of expert systems that can interrogate their environment via the use of analog-to-digital converters and other real-time sensor equipment. This capability for expert systems is expecially interesting in the areas of diagnostics and condition monitoring of turbomachinery. The expert reasoning capabilities of AI software, combined with automatic collection of vibration and other data, will result in vastly improved and earlier fault detection in rotating machinery in many industries such as electric utilities and aircraft maintenance. In addition, many of the same concepts can be applied to process monitoring and control applications.

The methods by which an expert system outputs its information to a user are important also. The possession of expert knowledge is useless if that knowledge cannot be effectively transmitted to a nonexpert. The standard printed textual output format has been shown to be a very poor and inefficient means of communicating the large volumes of complicated and technical information such as is possessed by expert systems. Pictorial and graphic displays have the capability of compacting information, thereby making it more readily understandable. Adding color to an animation to the graphics display further enhances the transmittal of information. Envision an expert system that is capable of showing an inexperienced maintenance technician a picture of a mechanical system, assembly or component, or showing an animated sequence that depicts the repair of an item. Hardware for graphics output of this kind is readily available for most computer systems.

Another possibility for information output is that of speech synthesis. An expert system could be used to verbally guide the maintenance technician through the repair process. Combine this with a speech recognition system, and the maintenance technician could indicate to the expert system when he is ready to proceed, or he could simply ask the expert for more detailed instructions without the need to break his concentration on the task being performed.

The objective of implementing expert systems that use alternative input/output techniques is to enhance and simplify the interface to user and to thus make the expert system more widely applicable. A key point to remember is that expert systems are the "experts" so that the user does not have to be an expert. The hardware to implement these interfaces is readily available. However, hardware for the special purpose LISP machines (LMI, Symbolics) is less common. The general purpose systems such as VAX and the micro-computers have supported such hardware for a long time in non-AI environments and have generally provided a LISP collatable interface. These considerations must be weighed, together with the AI development support offered by LISP Machines when choosing the computer hardware for an expert system.

C.5.0 NATURAL LANGUAGE INTERFACES

Natural language is perhaps the stickiest and most difficult area of AI system development. Programming the rules of grammar and massive vocabulary is not sufficient to reproduce the conceptual and contextual expressiveness and syntactical flexibility of human languages. This requires restricting any one NL interface to a specific application.

Natural language systems provide an interface between a person and a complex computer program that lets the user work with the human language he or she normally writes or speaks. Several programs using natural language techniques have been developed to give computer users a simple interface with data bases. Artificial Intelligence Corporation, Waltham, MA has developed the first real commercial application called Intellect, to help users specify quiries to various DBMSs that run on IBM systems.

This product allows the nontechnical user to access the corporate databases using conversational English. Without such help the expensive data bases set up in FORTRAN or COBOL can be accessed only by technical programmers. Computer scientists have long realized that for computers to reach their full potential, they have to be interfaced with natural languages such as English. They have been developing natural language interfaces which convert natural languages to computer languages. Currently, most products are designed to serve as front ends for databases. Users first input a request for information in English, which is converted by the natural language system into a formal database querry language. The relevant information is retrieved and displayed for the user.

The most advanced natural language systems today exhibit only a small fraction of the average human's capability to understand language.

Natural language systems exploit three linguistic fields: syntax, semantics, and pragmatics. The best understood of these is syntax, the rules used to combine words into phrases and sentences. Systems generally employ parsers to break sentences into their component parts and analyze them further. Some form of semantic analysis, which examines the meaning of words and their relationships, must be applied so that the most applicable way of parsing is chosen. A conceptual dependence representation which relies almost entirely on semantics is often used. When a word is characterized as a particular primitive act, it raises certain expectations in the system about its relationship with other words in the sentence. The system searches for confirmation of its expectations and analyzes sentences through several repetitions of this procedure, moving from word to word.

The use of semantic grammars, which mix semantic constraints with the syntactic patterns of work order, has also been advocated. However, these grammars need to be developed for each application.

Kaplan of Xerox Palo Alto Research Center developed lexical functional grammer (LFG), which uses the grammar of English and the semantic restrictions of a data base. An algorithm distributes the information about the semantics into the syntactic grammar. With this algorithm, production of grammars for specific data bases is automated.

Pragmatics, the third linguistic field in natural language systems has barely been touched by system developers. Pragmatics involves an interpretation of sentences that takes into account such variables as who wrote the sentence and where and when it was written. Such analysis requires the system to have extensive world knowledge in order to fully understand sentences.

The ASK system from Caltech runs on a Hewlett-Packard disktop computer. It provides both natural language capability and sophisticated data base management. This system has a highly optimized disk access scheme such that good response times are obtained.

THEMIS, a natural language system from Frey Associates, performs such functions as sorting, logical comparisons, and calculations. Themis understands statements even if they contain misspellings and typos. If words not in the system vocabulary are entered, those words can be taught to the system for an answer.

THEMIS, implemented in INTERLISP requires from 1.5 to 2 megabytes of a vax computer's main memory. It interfaces directly with the VAX data-base management system, DB MS/22, as well as with all standard VAX computer files. Themis can also interface with Data General corporation's version of Oracle, a relational data base management system.

INTELLECT acts as a hub between a number of different software systems. For example, a user might type, "show me a bar graph comparing our 1985 year-to-date sales with our 1985 estimated year-to-date sales by region". Intellect translates the request into the data base query language to obtain the necessary data and then interacts with graphics software, which transforms the rough data into a chart.

The INTELLECT natural-language processing system progresses through several steps to interpret and respond to a user's request for information. First, the scan function breaks the request "What were our New York sales in 1982?" into appropriate work entities, using information from both the computer's database and the system's lexicon (vocabulary). The scanner can't just assume that breaks occur at every space; for instance, "New York" should be classified as one, not two, word entities. Next, the parser grammatically diagrams the sentence in as many ways as possible and passes a list of partial interpretations to the weed function. Weed attempts to "fill in the holes" in the partial interpretations, using the database for guidance. Any complete interpretations that result pass to the Decide process. Decide, in part, uses the relative difficulty with which each full interpretation was produced to assign preference values to each. If the system can't clearly choose one final interpretation based on its perference rating and from additional searching of the database, it may query the user for additional information. For example, it might ask if "New York" refers to the city or the state, a decision it couldn't reach alone if its database contained sales Figures for each. Once the final interpretation is chosen, the system retrieves the requested information, processes and organizes it as necessary, and displays it to the user.

This system has abilities to interpret meanings in context. For example, in a user query, "compare January sales Figures for Chicago, New York City and not New York State". The system infers what the user means rather than what he says.

Gary Hendricks of Symantec developed a natural language query system for individuals who want to perform complex maneuvers with their databases without needing to know a programming language. The program was developed in INTERLISP and then translated to PASCAL.

Paul Martin is working on a system called TEAM at SRI. TEAM stands for Transportable English Database Access Medium and aims at natural language processing that allows a domain expert rather than an AI expert to move the system to a new application. The system comprises diamond parser constructor functions, the parse tree, semantic translators, basic pragmatic functions, scope determiners, and schema translator. The data base expert teaches TEAM the terms, concepts and relationships in his database. After that anyone can use TEAM to pose simple or complex questions about the database in natural English.

CLOUT developed by Microrim, Inc., Bellevue, Washington is designed for use with the IBM PC or compatible machines and microrim's relational data-base management systems.

Bobrow, et.al. developed GUS, (General Understanding System) an English dialog system. The system was interesting because of phenomena of natural dialog that it attempts to model and because of principles of program organization around which it was designed. GUS was able to respond appropriately in many cases where the client would seize the initiative and volunteer information that was not asked for or refusing to answer a question as asked. However, GUS never reached a stage where it could be turned loose on a completely naive client, however cooperative.

GUS represented a beginning step towards the construction of an intelligent language understanding system. Although GUS itself was not intelligent, it showed that an intelligent language understander must have a high quality parser, a reasoning component, and a well structured database of knowledge. The knowledge is of several types, from language specific information and expertise in the topic areas in which it can converse to a broad general knowledge of the world that must be used to interpret people's utterances. A robust natural language understanding system can cope with unrestricted natural language input by making plausible assumptions or by querying the user. If an expert system helps a naive user to communicate with a system through such robust, self-explanatory natural language interface, the need for training and human expertise diminishes. Thus, the functions provided by the expert system become truly "automated".

ACE (Academic Counseling Experiment) is a natural language test processing system under development at the University of Connecticut. ACE is intended to perform in real-time and in detail the tasks of a facility advisor. The system incorporates conceptual analyzer, APE (Academic Parsing Expert), and a conceptual generator CGEN. Both APE and CGEN communicate with the conversational and knowledge base management components of ACE using a variant of conceptual dependency (CD).

C.6 BIBLIOGRAPHY

Alexander, T., "Practical Uses for "Useless" Science", Fortune, May, 1982, pp.139-145.

Barr, A. and E. Feigenbaum, The Handbook of Artificial Intelligence, Vol. 2, William Kaufmann, Inc., 1982.

Bobrow, D.G., R.M. Kaplan, M. Kay, D.A. Norman, H.Thompson and T. Winograd, "GUS, A Frame-Driven Dialog System", Artificial Intelligence, Vol. 8, 1977, pp.155-173.

Buchanan, B. and E. Feigenbaum, "DENDRAL and META-DENDRAL: Their Applications Dimension", Artificial Intelligence, Vol. 2, 1978, pp.5-24.

Clark, K.L., and F.G. McCabe, "Micro-PROLOG: Programming in Logic", Prentice-Hall International Series in Computers, 1984.

Clocksin, W.F., and C.S. Mellish, "Programming in PROLOG". Springer-Verlag Berlin, 1981.

Cullingford, R.E., and M. Selfridge, "Real-World Natural Language Interfaces to Expert Systems", IEEE Paper CH 1887-9, 1983, pp.89-.

Davis, D.B., "English: The Newest Computer Language", High Technology, February 1984, pp.59-63.

Duda, R. and J. Gaschnig, "Knowledge-based Expert Systems Come of Age", Byte, September, 1981.

Duda, R., J. Gaschnig, and P. Hart, "Model Design in the Prospector Consultant System for Mineral Exploration", Expert Systems in the Microelectronic Age, Ed. D. Michie, Edinburgh University Press, 1979, pp.153-167.

Forgy, C.L., "OPS5 User's Manual", CMU-CS-81-135, July 1981, Department of Computer Science, Carnegie-Mellon University.

Green, J.O., "Making Computers Smarter", Popular Computing, January 1984, pp.97-106.

Hayes-Roth, F., D. Waterman, and D. Lenat, Building Expert Systems, Addison Wesley Publishing Company, Inc., 1983.

Hirsch, A., "Artificial Intelligence Comes of Age", Computers, & Electronics, March 1984, pp.63-67 & 93-96.

Manuel, T, and S. Evanczuk, "Commercial Products Begin to Emerge From Decades of Research", Electronics, November 3, 1983, pp.127-129.

McClellan, D.T., "LISP for Your Personal Computer", Computers and Electronics, March 1984, p.66.

"OPS5e User's Manual", Verac Incorporated, August, 1983

Rauch. H.E., "Probability Concepts for an Expert System Used for Data Fusion", The AI Magazine, Fall 1984, pp.55-60.

Rich, E., "Artificial Intelligence", McGraw-Hill Inc., 1983.

Teitelman, W., and L. Masinter, "The INTERLISP Programming Environment", Computer, April 1981, (IEEE), pp.25-33.

Wasserman, K., "Physical Object Representation and Generalization: A Survey of Programs for Semantic-Based Natural Language Processing", The AI Magazine Winter, 1985, pp.28-41.

Weiner, J., "Logic Programming and PROLOG", Computer & Electronics, Jan, 1985, pp.68-71.

Weiss, S., and C. Kulikowski, "A Practical Guide to Designing Expert Systems", Rowman & Allanheld, 1984.

Wilensky, R., "Talking to UNIX in English: An Overview of an On-Line UNIX Consultant, The AI Magazine, Spring 1984, pp.29-39.

Winston, P., and B.K.P. Horn, "LISP", Addison Wesley Publishing Co., 1981.

☆U.S.Government Printing Office: 1987 — 748-061/60712